In presenting the dissertation as a partial fulfillment of the requirements for an advanced degree from the Georgia Institute of Technology, I agree that the Library of the Institute shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish from, this dissertation may be granted by the professor under whose direction it was written, or, in his absence, by the Dean of the Graduate Division when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from, or publication of, this dissertation which involves potential financial gain will not be allowed without written permission.

3/17/65
b

SYNTHESIS OF NOISELESS COMPRESSION CODES

A THESIS

Presented to

The Faculty of the Graduate Division

by

Brian Parker Tunstall

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of Electrical Engineering

Georgia Institute of Technology

September, 1967

SYNTHESIS OF NOISELESS COMPRESSION CODES

Approved:

[redacted]

Date approved by Chairman: 10/10/67

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to Dr. R. P. Webb for his guidance and assistance during the development of this thesis. I also wish to thank Drs. A. M. Bush and R. H. Pettit for their services as members of the reading committee.

Special appreciation is given to the Schlumberger Well Surveying Corporation for a fellowship during the period when this research was conducted.

TABLE OF CONTENTS

TABLE OF CONTENTS (Concluded)

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Concluded)

SUMMARY

Two synthesis procedures are determined for digital compression codes. A compression code is a digital transformation having as its objective minimization of the expected value of the compression ratio, i.e., the ratio of the output digital sequence length to the input digital sequence length.

The system under consideration consists of a source, a coder, a channel, a decoder, and a user. The source is an ergodic, controllable-rate, first order, Markov random process. The coder is a reversible digital transformation. The channel is noiseless and transmits symbols of arbitrary durations. The decoder is the inverse of the coder transformation, so that the user receives exactly the same sequence emitted by the source.

A compression code is defined by two word sets, an input word set and an output word set. Encoding consists of dividing the input digital sequence into subsequences, each of which is a word in the input word set. Each subsequence causes the coder to emit the corresponding word from the output set. Thus an output sequence is formed.

Both word sets in the code are constrained to have the prefix property and to be complete. This implies that any input sequence can be divided into subsequences and coded into an output sequence uniquely and efficiently.

Three classes of codes are considered. If the input word set con-

tains words of uniform length, the code is Class A. If the output set is uniform, the code is Class B. If neither set is uniform, the code is Class C. If both word sets are arranged in numerical (alphabetic) order, the code is alphabetic. The size of a code is the number of words in either word set.

For each of the two synthesis procedures, a cost function is first determined. Extremization of the cost function yields a code. The first cost function is extremized by a combinatorial algorithm, the second by dynamic programming.

The first cost function is the limit, as the length of the input sequence goes to infinity, of the expected compression ratio. This limit is shown to be

$$G_1 = \frac{\sum_{i=1}^{N} p_i m_i}{\sum_{i=1}^{N} p_i \ell_i}$$

where p_i is the probability of occurrence of the i^{th} input word, ℓ_i is length of the i^{th} input word, and m_i is the length of the i^{th} output word.

A combinatorial algorithm has been found which synthesizes the optimum Class B code, i.e., the Class B code for which G_1 is less than for any other Class B code of the same size.

The algorithm begins with a basic word set containing the individual letters of the input alphabet. The input alphabet is assumed to contain m letters. The most probable word (letter) is removed from the basic set and replaced by m words, each of which is the removed word suffixed

by a letter from the input alphabet. Again the most probable word is suffixed, and this continues until the input set of the desired size is formed. The output set is, of course, just a set of uniform length words. This procedure is valid for zero memory sources, equal duration output letters, and any size alphabets.

With the same conditions on sources, channels, and alphabets, an optimum Class A code can be formed by the Huffman combinatorial algorithm for compact codes.

The efficiency of a compression code is the ratio of the output sequence information rate in bits per second to the channel capacity in bits per second. It is shown that for some sets of source statistics and code sizes the optimum Class A code is more efficient than the optimum Class B code, and that for some other sets it is not.

The cost function for the second synthesis procedure is based on a functional representation of a code. If a binary sequence, either finite or infinite in length, is preceded by a decimal point, the sequence assumes a numerical value contained in the unit interval. Therefore a code can be represented by a relation having for domain and range the unit interval. If both word sets are complete and have the prefix property and if the code is alphabetic, then the relation is a continuous, monotonically increasing function. This code function will have, in general, a countably infinite number of discontinuities in its first derivative.

A cost function based on the functional representation of a code is derived by again taking the limit, as the length of the input sequence goes to infinity, of the expected compression ratio. To obtain a tractable expression for this limit, the code function is piecewise linearly approxi-

mated. The breakpoint coordinates of the approximation are chosen to be the numerical values of the words in the input and output word sets. The resulting cost function is

$$G_2 = \sum_{i=1}^{N} (F(x_{i+1}) - F(x_i)) \log_2 \frac{x_{i+1} - x_i}{y_{i+1} - y_i}$$

where x_i is the numerical value of the i^{th} input code word, y_i is the numerical value of the i^{th} output code word, and $F(x)$ is the probability distribution function over the set of input sequence numerical values.

The cost function G_2 is separable and can, therefore, be minimized by dynamic programming. This reduces a single minimization of a function of 2N-2 variables to N - 1 minimizations of a function of two variables and makes functional synthesis computationally feasible. The value N is the code size.

Functional synthesis is flexible enough to include several interesting special cases. The distribution function, $F(x)$, is easily calculated for Markov sources. Consequently, functional synthesis is not limited to the case of zero memory sources as are the combinatorial procedures for Class A and B codes.

The value of each y_i can be non-linearly transformed to extend the validity of functional synthesis to the case of unequal duration output letters.

Finally, constraints can be applied to the values of x_i and y_i considered in the minimization which result in Class A codes, Class B codes, or codes with bounded maximum code word lengths in either word set.

A functionally synthesized code is not generally optimum because it is constrained to be alphabetic and because the cost function is based on an approximation to the code function. However, differences between efficiencies of functionally synthesized codes and optimum codes are shown by example to be negligible for a large range of code sizes and source statistics.

Therefore, the functional synthesis procedure presented is near-optimum. It is valid for Markov sources, output letters of arbitrary durations, bounded maximum code word lengths, and Classes A, B, or C. It is constrained to the case of alphabetic codes and binary output alphabets.

Functionally synthesized compression codes are applied to simple facsimile images. The effects of certain parameters of the images on compression are studied.

CHAPTER I

INTRODUCTION

The objective of this research is to determine synthesis methods for certain classes of noiseless compression codes. A compression code is a method of transforming one digital sequence into another digital sequence which is shorter. The transformation is reversible, and since the channel is assumed noiseless the receiver output sequence is identical to the source sequence.

The approach taken is to derive cost functions which are equal to the compression ratio and by extremizing the cost functions to determine optimum or near optimum compression codes. Two cost functions are derived which differ in the code parameters by which they are expressed. One of these cost functions is extremized by a combinatorial algorithm and the other by the application of dynamic programming. Two synthesis methods are thus determined and the features of each are investigated. Generalizations and special cases included are Markov sources, non-binary alphabets, constrained maximum code word lengths, non-equal cost code letters, and applications to facsimile.

Compression Codes

Economic operation of a digital data communication link requires a compact digital representation of the original message. The transformation which accomplishes this representation can usually be broken into

two successive transformations, the first being irreversible while the second is reversible. The irreversible transformation approximates the original message in a manner which satisfies a prescribed fidelity criterion. This type of transformation is usually called redundancy removal, source coding, or data compression.

The second or reversible transformation maps one digital sequence into another shorter sequence. This type of transformation is called compression coding.

A few examples of messages are television pictures, human speech, the output of a counter in a physical experiment, and the daily report of the New York Stock Exchange. If the original message is analog, it is assumed that the irreversible transformation includes the process of digitizing.

Consider specifically the example of human speech. A popular approach to processing speech for removing redundance is the channel vocoder. The analog speech signal is passed through a bank of contiguous bandpass filters. The filter outputs are rectified, lowpass filtered, and sequentially sampled and quantized. The resulting digital sequence cannot be transformed exactly back to the original speech signal; it is an approximation for which the fidelity criterion is intelligibility. Thus, the channel vocoder corresponds to the irreversible transformation discussed above.

Further processing on the vocoder output is desirable because the probability densities of filter outputs are neither uniform nor mutually independent. This further processing is usually a reversible digital code and corresponds to the reversible transformation discussed above.

There are many other examples of the application of irreversible and reversible transformations to typical messages.

In this research attention is confined to the problem of reversible compression codes. A block diagram of the system considered is given in Figure 1.

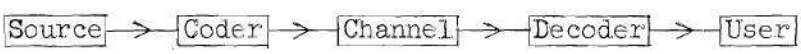


Figure 1. Compression Code System

In this diagram, the source represents either a digital message or an analog message followed by an irreversible transformation which includes digitization. This source is assumed to be an ergodic, digital, controllable rate, Markov random process. The output of the source is a sequence of letters drawn from the alphabet A, of size N_A. Ergodicity is assumed

$$A = \{\alpha_1, \alpha_2, \ldots, \alpha_{N_A}\} \tag{1-1}$$

because ensemble averaging is used in derivations below where time averages are actually the quantities of interest. Allowing the source rate to be controllable is a simple way to eliminate the buffer and memory considerations normally associated with data compression. The conditional probabilities of the source symbols are denoted by s_{ij}.

$$s_{ij} = \Pr\ (j^{th} \text{ alphabet letter occurs} \mid i^{th} \text{ alphabet letter has just occured}) \tag{1-2}$$

The unconditional probabilities are denoted by s_i.

$$s_i = \Pr\ (i^{th} \text{ alphabet letter occurs}) \tag{1-3}$$

Since the source is stationary (this is implied by ergodicity) and Markov, it is completely specified statistically by the set of conditional probabilities, s_{ij}.

The coder in Figure 1 transforms the source sequence into a coded sequence composed of letters drawn from an alphabet B, of size N_B.

$$B = \{\beta_1, \beta_2, \cdots, \beta_{N_B}\} \tag{1-4}$$

The channel is assumed to be noiseless and, in general, to require different amounts of time to transmit the various letters from alphabet B. The amount of time required to transmit β_i will be denoted by c_i. A simple example of unequal cost letters is the Morse code alphabet consisting of the dot and dash.

The decoder performs the inverse of the transformation performed in the coder, and the sequence presented to the user is identical to the output of the source.

The exact structure of a compression code will now be presented.

Code Structure

For the purposes of this discussion a word is defined as a row of letters which occur sequentially beginning with the leftmost letter.

A coder transformation, or code, is defined by two word sets, an

input set and an output set. An example of a code is given in Figure 2.

Input Set		Output Set
00	→	0
01	→	10
1	→	11

Figure 2. Code Example

Each word in the input set is a sequence of letters drawn from alphabet A, Equation (1-1); each word in the output set is a sequence of letters drawn from alphabet B, Equation (1-4). In this example, both alphabets are binary.

Encoding consists of dividing the input or source sequence into subsequences, each of which is a word from the code input word set. When the i^{th} input word is received from the source by the coder, the coder generates the i^{th} output word. Thus an output or channel sequence is formed. An example of an input sequence and the output sequence formed by the code in Figure 2 is given in Figure 3.

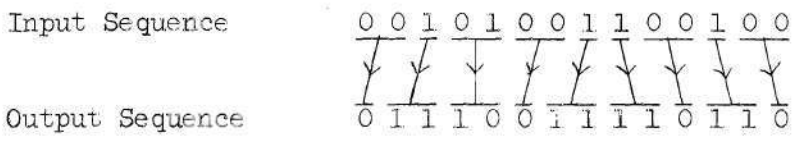


Figure 3. Encoding Example

This example shows how the input sequence is divided and how output words are generated.

A word set can always be associated with a tree graph. A tree graph is a linear graph which satisfies three properties:

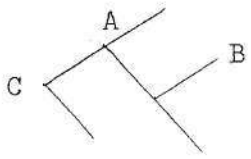
1) it forms no closed paths,

2) all but one of its nodes are the intersection of N+1 links, and

3) one node is the intersection of N links.

Here, N is the size of the alphabet used to form the word set.

An example of a tree graph with N = 2 and an associated word set is shown in Figure 4.



0 0

0 1 0

0 1 1

1

Figure 4. Tree Graph Example

A point such as B at the end of a branch is called a terminal node; a point such as A at the fork of branches is called an interior node. Point C, the node formed by the intersection of N links, is called the root node.

Each word in the associated word set corresponds to one node. The word assigned to a particular node (either interior or terminal) can be determined by recording a zero for each upward fork and a one for each downward fork along a path from the root node to the node in question.

The four words in the right of Figure 4 correspond to the four terminal nodes of the tree graph.

Two important properties of word sets are now defined.

A word set is complete if every terminal node of the associated tree is assigned a word; a word set is proper if no word is assigned to an interior node. The word set shown in Figure 4 is proper and complete. If any one word were removed from the set it would be incomplete. If the word 01 were added to the set it would be improper. A word set which is proper is sometimes said to have the prefix property, since no word is the prefix of any other word in the set.

In order for every possible input sequence to the coder to be divisible into a sequence of input words, the input word set must obviously be complete. For this division to be unique the word set must be proper.

As is shown later, an incomplete output word set can always be made more efficient by reducing the lengths of one or more words until the set is complete.

It has been shown using the McMillan inequality that there exists a proper output word set which is as efficient as any output word set, proper or improper (1).

Therefore, in this dissertation, consideration has been limited to the synthesis of codes which have complete and proper word sets for both input and output. A code can be partially classified by stating its size and whether or not it is alphabetic.

The size of a code is the number of words in either of its word sets.

A word set is said to be alphabetic if its words are ordered ac-

cording to increasing numerical value. In determining numerical value, the alphabet from which the letters are drawn is assumed to be monotonically increasing in numerical value, and the leftmost letter, or digit, in each word is assumed to be preceded by a decimal point.

If placing the input word set of a code into alphabetic order also alphabetizes the output set, the code is said to be alphabetic.

Three classes of compression codes are defined in the next section.

Code Classes

Consider the following examples of codes in Figure 5. Code 1 is alphabetic: both word sets are in alphabetic order. Code 3 is, likewise, alphabetic. However, the output word set of Code 2 is alphabetic while the input word set is not. Therefore, Code 2 is non-alphabetic.

By observation, the sizes of Codes 1, 2, and 3 are four, four, and five, respectively.

The words of the input word set of Code 1 are all the same length. Codes which satisfy this constraint are Class A codes. The words of the output word set of Code 2 are all the same length. Codes which satisfy this constraint are Class B codes. Codes which satisfy neither of these constraints, such as Code 3, are Class C codes.

Classes A, B, and C codes constitute a very general set of compression codes. In fact, most of the research in compression codes that is available in the literature is concerned with special cases of the system model and code classes described above. Some of the more important research in this area is reviewed in the next section.

0 0	→	0
0 1	→	1 0
1 0	→	1 1 0
0 0	→	1 1 1

Code 1: Class A, Size 4, Alphabetic

1	→	0 0
0 1	→	0 1
0 0 0	→	1 0
0 0 1	→	1 1

Code 2: Class B, Size 4, Non-alphabetic

0 0 0	→	0
0 0 1	→	1 0
0 1	→	1 1 0 0
1 0	→	1 1 0 1
1 1	→	1 1 1

Code 3: Class C, Size 5, Alphabetic

Figure 5. Code Examples

Review of Compression Coding Research

Some of the codes described below are said to be optimum. This means they yield as much data compression as any other code of the same size and class for a given special case. The constraints for the special cases might be, for example, equal cost output letters, alphabetic codes, or zero memory sources. The constraints are carefully described in each case.

A classic paper in the compression code literature was written by Huffman in 1952 (2). He demonstrated and proved the optimality of a synthesis method for Class A codes using a combinatorial approach. His method is valid only for a zero memory source and for equal cost output letters. This method is discussed in Chapter II.

In 1959, Gilbert and Moore presented an optimum alphabetic Class A synthesis procedure (3). This procedure was valid for equal cost output letters and a zero memory source. Their approach was also combinatorial.

Karp, in 1961, presented an optimum Class A synthesis procedure using an integer programming approach (4). It was constrained to zero memory sources but was valid for unequal cost output letters.

Billingsley, in 1961, used a functional representation of coding to prove Shannon's Noiseless Coding Theorem (5). This did not result directly in a synthesis procedure, but the functional concept is the basis for the synthesis procedure presented in Chapter III of this dissertation.

Marcus (6), in 1957, and Laemmel (7), in 1958, both presented suboptimal synthesis procedures for Classes B and C codes.

Blachman, in 1954, described a suboptimal procedure for Class A

codes which was valid for unequal cost output letters (8).

In 1966, Golomb discussed a synthesis procedure for Class C codes which was optimum for a limited class of zero memory sources (9).

In summary, synthesis procedures for optimum Class A codes exist which are valid for zero memory sources and unequal cost output letters or for zero memory sources and alphabetic codes; no synthesis procedures for optimum Class B or Class C codes have previously been developed.

Research Results

The results of this dissertation are two synthesis procedures. The first is a synthesis procedure for optimum Class B codes. It is valid for zero memory sources, equal cost output letters, and any size input and output alphabets. This procedure, which is based on a combinatorial approach, is discussed in Chapter II.

The second is a synthesis procedure for suboptimal Class A, B, or C codes. It is valid for Markov sources and unequal cost output letters. It is constrained to the case of alphabetic codes and binary alphabets. This procedure, which is based on a functional representation of a code, is introduced in Chapter III.

CHAPTER II

COMBINATORIAL CODES

In the discussions following, a combinatorial code is defined as a code for which the synthesis procedure is a combinatorial algorithm and for which combinatorial analysis can be used to prove optimality. Combinatorial methods for Classes A, B, and C codes are now discussed.

Cost Function

The most important property of a compression code is its ability to produce, on the average, the shortest possible output sequence for a given length input sequence. Other properties, such as synchronizability and bounded delay, are not considered here.

Comparison of codes to determine optimality requires a figure of merit, or cost function, which is a monotone function of compression. Synthesis of the optimum compression code is then reduced to choosing the code which extremizes the cost function. Since the compression referred to above is an average or expected quantity, the cost function will depend on source statistics as well as code structure.

The compression ratio, G_Q, is defined as

$$G_Q = \frac{R}{Q} \tag{2-1}$$

where

$$R = \text{output sequence length} \tag{2-2}$$

and

$$Q = \text{input sequence length} \tag{2-3}$$

Both R and G_Q are random variables; Q is specified.

An ambiguity arises because of the finite Q: what happens if the uncoded remainder of the input sequence after a certain number of encoding operations is a fraction of an input code word, i.e., the division of the input sequence does not require an integral number of input code words?

To avoid this ambiguity the following rule is used for coding finite length input sequences: When the uncoded remainder of the input sequence is shorter than the longest word in the input word set, which has length ℓ_{max}, the remainder is transmitted in uncoded form.

It is assumed that the decoder at the receiver knows Q and ℓ_{max} so that no special indication for the uncoded remainder is needed.

The compression ratio can then be written

$$G_Q = \frac{R}{Q} = \frac{\sum_{i=1}^{N} v_i m_i + Z}{\sum_{i=1}^{N} v_i \ell_i + Z} \tag{2-4}$$

where

v_i = number of occurrences of i^{th} input word (2-5)

m_i = length of i^{th} output word (2-6)

ℓ_i = length of i^{th} input word (2-7)

Z = length of remainder (2-8)

In Equation (2-4), G_Q, R, $\{v_i\}$, and Z are all random variables, while Q,

$\{m_i\}$, and $\{\ell_i\}$ are assumed known.

For given source statistics and a particular code, the expected value of G_Q can be calculated. Repeating this calculation for all possible codes allows the optimum code to be identified. This procedure for identifying a code is referred to as enumeration.

Enumeration using the compression ratio, G_Q, is unsatisfactory for two reasons. The first reason is that the amount of computation required for enumeration tends to infinity as Q, the input sequence length, tends to infinity.

The second reason is that the amount of computation required for a code of size N is, in the most general case, proportional to $((N-1)!)^2$. This proportionality is shown in Chapter III.

A simple expression for the asymptotic value of G_Q as Q goes to infinity is now derived. It will be shown that this eliminates the problem associated with input sequence length, Q, and in some cases reduces the problem associated with code size, N.

Let

$$G_Q = \frac{R}{Q} = \frac{\sum_{i=1}^{N} v_i m_i + Z}{\sum_{i=1}^{N} v_i \ell_i + Z} \qquad (2\text{-}9)$$

$$= \frac{\sum_{i=1}^{N} \frac{v_i}{n} m_i + \frac{Z}{n}}{\sum_{i=1}^{N} \frac{v_i}{n} \ell_i + \frac{Z}{n}}$$

where

$$n = \sum_{i=1}^{N} v_i \tag{2-10}$$

is the number of complete words in the input sequence.

Assuming a zero memory source, the law of large numbers (10) states

$$\lim_{Q \to \infty} \Pr\left[\left| \frac{v_i}{n} - P_i \right| < \epsilon\right] = \lim_{n \to \infty} \Pr\left[\left| \frac{v_i}{n} - P_i \right| < \epsilon\right] = 1 \tag{2-11}$$

where P_i is the probability of the i^{th} input word and $\epsilon > 0$.

Therefore

$$E[G_Q] \geq \frac{\sum_{i=1}^{N} (P_i - \epsilon)\, m_i + \frac{Z}{n}}{\sum_{i=1}^{N} (P_i + \epsilon)\, \ell_i + \frac{Z}{n}} \cdot \Pr\left[\left| \frac{v_i}{n} - P_i \right| < \epsilon\right] \tag{2-12}$$

$$+ \frac{m_{min} + \frac{Z}{n}}{\ell_{max} + \frac{Z}{n}} \cdot \Pr\left[\left| \frac{v_i}{n} - P_i \right| \geq \epsilon\right]$$

where

m_{min} = length of shortest output word

ℓ_{max} = length of longest input word

Similarly

$$E[G_Q] \leq \frac{\sum_{i=1}^{N} (P_i + \epsilon)\, m_i + \frac{Z}{n}}{\sum_{i=1}^{N} (P_i - \epsilon)\, \ell_i + \frac{Z}{n}} \cdot \Pr\left[\left| \frac{v_i}{n} - P_i \right| < \epsilon\right] \qquad (2\text{-}13)$$

$$+ \frac{m_{max} + \frac{Z}{n}}{\ell_{min} + \frac{Z}{n}} \cdot \Pr\left[\left| \frac{v_i}{n} - P_i \right| \geq \epsilon\right]$$

where

m_{max} = length of longest output word

ℓ_{min} = length of shortest input word

If n and Q tend to infinity and ϵ is arbitrarily small then by use of Equation (2-11) it can be shown that both the lower and upper bounds of $E(G_Q)$, Equations (2-12) and (2-13), tend to the same limit. Therefore, $E(G_Q)$ also approaches this limit.

$$\lim_{Q \to \infty} E[G_Q] = G = \frac{\sum_{i=1}^{N} P_i\, m_i}{\sum_{i=1}^{N} P_i\, \ell_i} \qquad (2\text{-}14)$$

This limit is defined as the cost function, G. The expectation in Equation (2-14) is with respect to the source random process which is assumed to be ergodic and zero memory and to produce infinite length sequences. As can be seen, the cost function, G, is just the ratio of the average output word length to the average input word length.

An optimum code for a zero memory source will now be defined as that code among all possible codes of a given size for which the cost

function, G, is minimum.

Synthesis of Class A Codes

Class A codes are constrained to have uniform length input words. The cost function for a Class A code can be simplified as shown in Equation (2-15)

$$G = \frac{\sum_{i=1}^{N} P_i \, m_i}{\ell} \tag{2-15}$$

where ℓ is the common length of all input words. The synthesis problem is reduced to choosing the output word set which minimizes the average output word length, M.

$$M = \sum_{i=1}^{N} P_i \, m_i \tag{2-16}$$

Huffman, in 1952, presented a systematic procedure for determining the output word set which minimizes M, i.e., he presented an optimum Class A synthesis procedure (2). Huffman's procedure and the proof of its optimality are contained in most introductory texts on information theory. For this reason it will not be repeated here. Huffman's procedure is valid for zero memory sources and equal cost output letters. The efficiencies of Class A codes synthesized by Huffman's algorithm are presented in a later section of this chapter for comparison with optimum Class B codes.

Synthesis of Class B Codes

Class B codes are constrained to have uniform length output words. The cost function for Class B codes can be written as shown in Equation (2-17).

$$G = \frac{m}{\sum_{i=1}^{N} P_i \ell_i} \tag{2-17}$$

where m is the common length of all the output words.

Thus the problem of optimum code synthesis is reduced to forming an input word set which maximizes

$$L = \sum_{i=1}^{N} P_i \ell_i \tag{2-18}$$

where L is the average input word length.

One result of this research is a systematic procedure for forming an input word set which maximizes L.

Assume a zero memory source produces symbols from an alphabet, A

$$A = \{\alpha_1, \alpha_2, \ldots, \alpha_{N_A}\} \tag{2-19}$$

having probabilities

$$S = \{S_1, S_2, \ldots, S_{N_A}\} \tag{2-20}$$

respectively. A basic word set is formed by listing the letters of the

input alphabet as shown in Figure 6.

$$\alpha_1$$
$$\vdots$$
$$\alpha_{N_A}$$

Figure 6. Basic Word Set

Assume the j^{th} word is removed from the set shown in Figure 6 and replaced by N_A words each of which is α_j followed by a letter of the alphabet. This results in a modified word set as shown in Figure 7.

This modification is a suffixing operation and results in the modified set of probabilities, P, shown in Equation (2-21).

$$P = \{S_1, \cdots, S_{j-1}, S_jS_1, S_jS_2, \cdots \quad (2\text{-}21)$$
$$\cdots, S_jS_{N_A}, S_{j+1}, \cdots, S_{N_A}\}$$

Any complete and proper word set will have size

$$N = N_A + k(N_A - 1) \quad (2\text{-}22)$$

for

$$k = 0, 1, 2, \ldots.$$

α_1

.

.

.

α_{j-1}

$\alpha_j \alpha_1$

$\alpha_j \alpha_2$

.

.

.

$\alpha_j \alpha_{N_A}$

α_{j+1}

.

.

.

α_{N_A}

Figure 7. Modified Word Set

This is because the tree graph associated with the basic word set has N_A terminal nodes; each suffixing operation changes one terminal node into an interior node and produces N_A new terminal nodes. Obviously, any complete and proper word set of size $N_A + K(N_A - 1)$ can be generated by K successive suffixing operations applied to the basic word set.

If the word chosen at each step to be removed and suffixed is the most probable word in its set, the procedure is called most probable word suffixing.

It is now proved by induction that the input word set which maximizes the average input word length, Equation (2-18), can be generated by most probable word suffixing.

Theorem 1: Most probable word suffixing generates the Class B code input word set which is optimum, i.e., which maximizes L.

Proof:

Assume the code size is

$$N = N_A + K(N_A - 1)$$

where K is a non-negative integer. If the i^{th} word in the word set after k suffixing operations has probability $P_i^{(k)}$ and length $\ell_i^{(k)}$, then L can be written

$$L = \sum_{i=1}^{N} P_i^{(K)} \ell_i^{(K)} \qquad (2\text{-}23)$$

The word chosen from the word set after k suffixing operations to be re-

moved and suffixed will be denoted by the index j_k. If the average input word length of the input word set formed after k suffixing operations is denoted by L_k, then Equation (2-23) can be rewritten as

$$L_K = \sum_{i=1}^{N_A+K(N_A-1)} P_i^{(K)} \ell_i^{(K)} \qquad (2\text{-}24)$$

$$= \sum_{\substack{i=1 \\ i\neq j_{K-1}}}^{N_A+(K-1)(N_A-1)} P_i^{(K-1)} \ell_i^{(K-1)} + \sum_{i=1}^{N_A} P_{j_{K-1}}^{(K-1)} S_i \left(\ell_{j_{K-1}}^{(K-1)} + 1 \right)$$

$$= \sum_{\substack{i=1 \\ i\neq j_{K-1}}}^{N_A+(K-1)(N_A-1)} P_i^{(K-1)} \ell_i^{(K-1)} + P_{j_{K-1}}^{(K-1)} \left(\ell_{j_{K-1}}^{(K-1)} + 1 \right)$$

$$= \sum_{i=1}^{N_A+(K-1)(N_A-1)} P_i^{(K-1)} \ell_i^{(K-1)} + P_{j_{K-1}}^{(K-1)}$$

$$= L_{K-1} + P_{j_{K-1}}^{(K-1)}$$

$$= L_{K-2} + P_{j_{K-2}}^{(K-2)} + P_{j_{K-1}}^{(K-1)}$$

$$= L_0 + P_{j_0}^{(0)} + P_{j_1}^{(1)} + \ldots + P_{j_{K-1}}^{(K-1)}$$

But

$$L_0 = \sum_{i=1}^{N_A} P_i^{(0)} \ell_i^{(0)} \qquad (2\text{-}25)$$

$$= \sum_{i=1}^{N_A} S_i = 1$$

Therefore it follows that

$$L_K = 1 + P_{j_0}^{(0)} + P_{j_1}^{(1)} + \ldots + P_{j_{K-1}}^{(K-1)} \tag{2-26}$$

It is now shown by induction that for any complete proper word set of size $N_A + K(N_A - 1)$, the value of L_K is maximized by most probable word suffixing. For notational convenience it will be assumed that after each suffixing operation, the words are reindexed in order of decreasing probability. Assume a word set of size $N_A + (K - 1)(N_A - 1)$ is given. The optimum choice for j_{K-1} is obviously

$$j_{K-1} = 1 \tag{2-27}$$

i.e., suffix the most probable word.

Now assume the optimum value of j_i is known to be

$$j_i = 1 \tag{2-28}$$

for all i greater than k

$$i = k + 1,\ k + 2,\ \ldots,\ K - 1 \tag{2-29}$$

It will be shown that the optimum value of j_k is

$$j_k = 1 \tag{2-30}$$

The values of $p_{j_0}^{(0)}$, $p_{j_1}^{(1)}$, ... , $p_{j_{k-1}}^{(k-1)}$ are independent of the value selected for j_k. Therefore, when choosing the optimum j_k, the objective is to maximize the partial sum, D, given by

$$D = p_{j_k}^{(k)} + p_1^{(k+1)} + p_1^{(k+2)} + \ldots + p_1^{(k-1)} \tag{2-31}$$

The maximization is performed over j_k. The superscripts "^" and "~" are used to denote variables which occur when choosing $j_k \neq 1$ and $j_k = 1$, respectively.

When a word having probability $p_{j_i}^{(i)}$ is suffixed it generates N_A new words each having probability less than $p_{j_i}^{(i)}$. Thus

$$p_1^{(k+1)} \geq p_1^{(k+2)} \geq \ldots \geq p_1^{(K-1)} \tag{2-32}$$

assuming Equations (2-28) and (2-29). Now assume $j_k \neq 1$. There are two possible cases:

1) $$\hat{p}_{j_k}^{(k)} < \tilde{p}_1^{(K-1)} \tag{2-33}$$

or

2) $$\hat{p}_{j_k}^{(k)} \geq \tilde{p}_1^{(K-1)} \tag{2-34}$$

Consider case 1). In case 1)

$$\hat{D} = \hat{p}_{j_k}^{(k)} + \hat{p}_1^{(k+1)} + \hat{p}_1^{(k+2)} + \ldots + \hat{p}_1^{(K-1)} \tag{2-35}$$

Obviously

$$\hat{p}_1^{(k+1)} = p_1^{(k)}$$

And, in general

$$\hat{p}_1^{(k+i)} = \tilde{p}_1^{(k+i-1)}, \quad i = 1, 2, \ldots, K-k-1 \tag{2-36}$$

Therefore

$$\hat{D} = \hat{p}_{j_k}^{(k)} + p_1^{(k)} + \tilde{p}_1^{(k+1)} + \tilde{p}_1^{(k+2)} + \ldots + \tilde{p}_1^{(K-2)} \tag{2-37}$$

$$< p_1^{(k)} + \tilde{p}_1^{(k+1)} + \tilde{p}_1^{(k+2)} + \ldots + \tilde{p}_1^{(K-1)} = \tilde{D}$$

since

$$\hat{p}_{j_k}^{(k)} < \tilde{p}_1^{(K-1)} \tag{2-38}$$

Now consider case 2). There exists some i

$$k + 1 \leq i \leq K - 1 \tag{2-39}$$

for which

$$\tilde{p}_1^{(i)} = \hat{p}_{j_k}^{(k)} \tag{2-40}$$

It follows that

$$\hat{D} = \hat{p}_{j_k}^{(k)} + \hat{p}_1^{(k+1)} + \hat{p}_1^{(k+2)} + \ldots + \hat{p}_1^{(i-1)} + \hat{p}_1^{(i)} \tag{2-41}$$

$$+ \hat{p}_1^{(i+1)} + \ldots + \hat{p}_1^{(K-1)} = \hat{p}_{j_k}^{(k)} + p_1^{(k)} + \tilde{p}_1^{(k+1)} + \ldots$$

$$+ \tilde{p}_1^{(i-2)} + \tilde{p}_1^{(i-1)} + \tilde{p}_1^{(i+1)} + \ldots + \tilde{p}_1^{(K-1)} = \tilde{D}$$

Therefore, considering both cases

$$\tilde{D} \geq \hat{D} \tag{2-42}$$

Consequently $j_k = 1$ is optimum and the proof follows by induction.

Q. E. D.

Recall that this procedure for optimum Class B codes is valid for any size input or output alphabet, but is constrained to zero memory sources and equal cost output letters. It is not necessary for the size of a Class B code to be an integral power of N_B, the output alphabet size. However, when it is not, the code is not as efficient as it could be if some of the output words were shortened. This would, in general, change the code into Class C.

The input word set must be complete so that any input sequence can be encoded. For this reason, any Class A code must have a size which is an integral power of N_A.

A comparison of the efficiencies of Class A and Class B codes having integral power of two sizes is presented in the next section.

Comparison of Class A and Class B Codes

One measure of the effectiveness of a compression code is its expected compression ratio. This is the specific quantity which is optimized by the synthesis techniques in this research. However, for tabulating results and comparing codes another measure of effectiveness is more convenient. This measure is the ratio of the coder output information rate in bits per second to the channel capacity in bits per second. This ratio

is always some fraction between zero and one and is referred to as the code efficiency.

The efficiencies of optimum Class A and Class B codes are shown in Figure 8. The codes were formed by Huffman's procedure or the procedure presented in the preceding section. Codes of sizes four and eight were synthesized for zero memory binary sources having entropies of .1, .5, and .9.

The probability of zero in a zero memory binary source having entropy of .1, .5, or .9 is .013, .11, or .316, respectively. As can be seen the Class B code is less efficient than the Class A code for all size and entropy combinations illustrated. However, for a probability of zero equal to .1 and a size of 32, the optimum Class B code is slightly more efficient than the optimum Class A code. Thus, it is not generally true that Class A codes are more efficient than Class B codes. In addition, the uniform output word length of Class B may be desirable for reasons other than compression, such as synchronizability.

The existence of combinatorial synthesis procedures for optimum Class A and Class B codes leads to a consideration of combinatorial procedures for Class C codes.

Combinatorial Class C Synthesis

No combinatorial synthesis procedure for optimum Class C codes has been found, even for the simplest case of binary alphabets, zero memory sources, and equal cost output letters.

It can be shown by counter example that the optimum input word set for a particular size of Class C code is not, in general, the optimum input

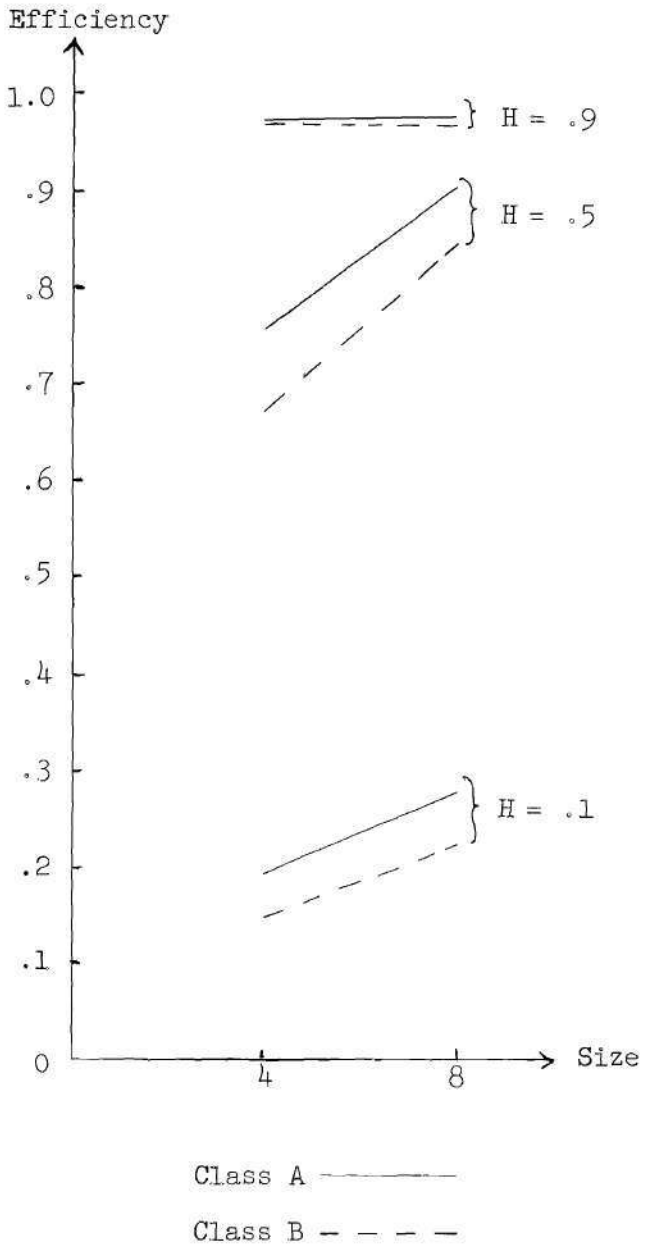


Figure 8. Optimum Class A and B Code Efficiencies

word set of a smaller size of Class C code modified by suffixing operations. Thus, a synthesis procedure for optimum Class C codes could not be a simple, repeated operation as in the case of Class B.

For this reason it is necessary that a different cost function be determined. This cost function is based on a functional representation of a code and is presented in the next chapter.

CHAPTER III

FUNCTIONAL CODES

A binary digital sequence, either finite or infinite, preceded by a decimal point becomes the binary decimal expansion of a point in the unit interval. Hence, a compression code can be considered to be a mapping from the unit interval to the unit interval. It can be shown that if both the input and output word sets are complete and proper, if the code is alphabetic, and if the input sequence has infinite length, then the mapping will be a continuous, monotonically increasing function for which both domain and range are the unit interval (11). Any code synthesized by a procedure based on its functional representation is called a functional code. Functional Class C codes are now considered.

Billingsley's Results

Under the conditions stated above, a code can be represented by a continuous monotonically increasing function from the unit interval into the unit interval. If the source statistics are known, the probability distribution function over the input point set can be determined. It, too, will be a continuous, monotonically increasing function from the unit interval to the unit interval. An example of a code function and a distribution function are given in Figure 9. In this figure, x denotes the input sequence numerical value, Φ the code function, and F the distribution function.

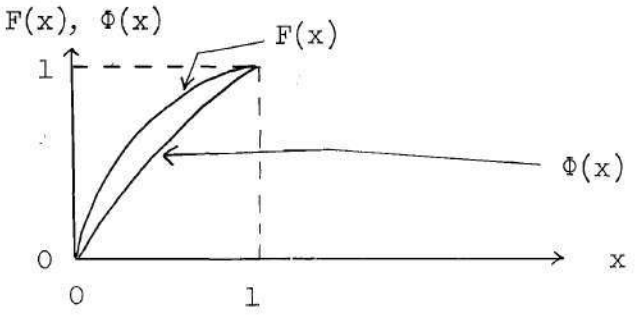


Figure 9. Code and Distribution Functions

In 1961, Billingsley (5) showed that for optimum compression the code should be chosen so that the code function, $\Phi(x)$, is identical to the distribution function, $F(x)$. More specifically, he found:

1) If $\Phi(x) = F(x)$, then the measure of the set of points, x, for which the efficiency is unity is one.

2) For any $\Phi(x)$, the measure of the set of points for which the efficiency is greater than unity is zero.

Thus any code for which $\Phi(x)$ equals $F(x)$ is optimum. However, choosing a code for which $\Phi(x)$ equals $F(x)$ will, in general, require an infinite sized code. Billingsley's results do not yield a synthesis procedure for finite sized codes. In this chapter a cost function based on functional representation is derived for finite sized codes. The expression for the cost function is a weighted average of points on the code function. Before the derivation of this expression is presented some basic theorems concerning binary expansions are proved.

Binary Expansions

Theorem 2, Theorem 3, and Theorem 4, which follow, are useful in

subsequent discussions of functional codes.

Let the i^{th} word in a complete and proper set of N alphabetized words, $\{\gamma_i\}$, have length ℓ_i. The numerical value of this word when preceded by a decimal point is denoted by x_i. An example of a word set and its associated numerical values and lengths is given in Figure 10.

$\gamma_1 = 0\ 0$	$x_1 = 0$	$\ell_1 = 2$
$\gamma_2 = 0\ 1$	$x_2 = 1/4$	$\ell_2 = 2$
$\gamma_3 = 1$	$x_3 = 1/2$	$\ell_3 = 1$

Figure 10. Word Set

As illustrated in Figure 10, lower case Greek letters are used in this thesis to represent actual digital symbols or sequences.

Theorem 2:

A necessary condition for the set $\{x_i\}$ to represent the words of a complete proper alphabetic word set of size N having lengths $\{\ell_i\}$ is

$$x_{k+1} - x_k = 2^{-\ell_k} \qquad (3\text{-}1)$$

$$k = 1, 2, \ldots, N$$

where

$$x_{N+1} = 1$$

Proof:

Since the x_i are labeled in monotonically increasing order, adja-

cent words will always have the following binary structure

$$\gamma_{k+1} = \eta \ \ell \ \mu_i \tag{3-2}$$

and

$$\gamma_k = \eta \ 0 \ \zeta_j \tag{3-3}$$

where

η = binary sequence of length m

μ_i = sequence of i zeros

ζ_j = sequence of j ones

and m, i, and j are non-negative integers.

Then

$$x_{k+1} - x_k = 2^{-(m+1)} \cdot 2^{-j} \tag{3-4}$$

$$= 2^{-(m+1+j)}$$

$$= 2^{-\ell_k}$$

for

$$k = 1, 2, \ldots, N - 1$$

By definition x_{N+1} equals one.

Obviously

$$\gamma_N = \zeta_j, \qquad j \geq 1$$

$$x_{N+1} - x_N = 2^{-j} = 2^{-\ell_N} \tag{3-5}$$

$$x_{k+1} - x_k = 2^{-\ell_k} \tag{3-6}$$

$$k = 1, 2, \ldots, N$$

Q. E. D.

Corollary 1: A necessary condition for a CPBA (complete, proper, binary, alphabetic) word set to have a maximum word length of ℓ_{max} is

$$x_{k+1} - x_k \geq 2^{-\ell_{max}} \qquad (3\text{-}7)$$

$$k = 1, 2, \ldots, N$$

where

$$x_{N+1} = 1$$

Theorem 3:

The longest word in a CPBA word set of size N has length, ℓ_{max}, bounded by

$$\ell_{max} \leq N - 1 \qquad (3\text{-}8)$$

Proof:

A binary suffixing operation adds one to the size of a word set. Any CPBA word set of size N can thus be formed with N-2 suffixing operations beginning with the binary alphabet. A single suffixing operation can increase the length of a word in a set by no more than one. Hence, the maximum length of any word after N-2 suffixing operations is N-1.

Q. E. D.

Theorem 4:

Necessary and sufficient conditions for a set $\{x_i\}$ to represent a CPBA word set of size N are

1) $$x_{k+1} - x_k = 2^{m_k+1-N} \qquad (3\text{-}9)$$

$$k = 1, 2, \ldots, N$$

and m_k is an integer satisfying

$$0 \leq m_k \leq N - 2$$

$$k = 1, 2, \ldots, N$$

and

$$x_{N+1} = 1$$

2)
$$x_{k+1} - x_k \leq 2^{-\hat{\ell}(x_k)} \tag{3-10}$$

$$k = 1, 2, \ldots, N$$

Here, $\hat{\ell}(x_k)$ is the length of the shortest binary sequence which has a numerical value of x_k.

Proof:

Necessity: Assume $\{x_i\}$ is the set of numerical values of the words in a CPBA word set. Then by Theorems 2 and 3, and the fact that the minimum word length in any CPBA word set is one

$$1 \leq \ell_i \leq N - 1, \qquad i = 1, 2, \ldots, N$$

Thus

$$x_{k+1} - x_k = 2^{-\ell_k} = 2^{-(m_k+1-N)} \tag{3-11}$$

$$k = 1, 2, \ldots, N$$

and each m_k is an integer satisfying

$$0 \leq m_k \leq N - 2$$

This proves the necessity of 1). By definition

$$\ell_i \geq \hat{\ell}(x_i)$$

Hence

$$x_{k+1} - x_k = 2^{-\ell_k} \leq 2^{-\hat{\ell}(x_k)} \tag{3-12}$$

$$k = 1, 2, \ldots, N$$

This proves the necessity of 2).

Sufficiency: Assume $\{x_i\}$ satisfies conditions 1) and 2). Choose a set $\{\ell_i\}$ such that

$$\ell_i = -\log_2 (x_{i+1} - x_i) \tag{3-13}$$

$$i = 1, 2, \ldots, N$$

By assumption 1) the set $\{\ell_i\}$ is a set of integers. It can easily be shown that the set $\{\ell_i\}$ satisfies the Kraft inequality (1).

$$\sum_{i=1}^{N} 2^{-\ell_i} = \sum_{i=1}^{N} (x_{i+1} - x_i) \tag{3-14}$$

$$= x_{N+1} - x_1$$

$$= 1 - 0$$

$$= 0$$

Therefore there exists at least one CPBA word set having lengths $\{\ell_i\}$.

It has not been shown that a word set having the lengths $\{\ell_i\}$ can have word values equal to $\{x_i\}$. However, by assumption 2)

$$\ell_i \geq \hat{\ell}(x_i), \qquad i = 1, 2, \ldots, N$$

Q. E. D.

Functional Cost Function

A cost function based on a functional representation of a code can now be derived.

Assume a binary source emits a finite length sequence of length Q which is encoded into a sequence of length R. It is again assumed that when the uncoded remainder of the input sequence is shorter than the longest word in the coder input word set, the remainder is transmitted unencoded. The expected output sequence length can be written

$$E[R] = \sum_{i=1}^{2^Q} u_i R_i \tag{3-15}$$

where

u_i = probability of i^{th} input sequence

and

R_i = length of i^{th} output sequence

According to Theorem 2, Equation (3-15) can be rewritten as

$$E[R] = -\sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (r_{i+1} - r_i) \tag{3-16}$$

where

$$F(q_i) = \Pr\ (\text{input sequence numerical value} < q_i)$$

q_i = numerical value of i^{th} input sequence

and

r_i = numerical value of i^{th} output sequence.

The expected value of the compression ratio is then

$$E[\text{Compression Ratio}] = E\left[\frac{R}{Q}\right] \tag{3-17}$$

$$= \frac{E[R]}{Q}$$

$$= -\frac{1}{Q} \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (r_{i+1} - r_i)$$

At this point the difficulty is again encountered that the amount of computation required for enumeration is exponentially dependent upon the input sequence length. To circumvent this difficulty the limit of the expected compression ratio as Q goes to infinity will be taken. In the appendix it is shown that the limit is

$$\lim_{Q \to \infty} \frac{E[R]}{Q} = -\lim_{Q \to \infty} \frac{1}{Q} \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \tag{3-18}$$

$$\cdot \log_2 (\Phi(q_{i+1}) - \Phi(q_i))$$

where $\Phi(q)$ is the code function, i.e., the mapping of the infinite length input sequence of value q into an infinite length output sequence of value

$\Phi(q)$. Hence, by the basic property of limits, for sufficiently large Q the code which minimizes G_Q'

$$G_Q' = - \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (\Phi(q_{i+1}) - \Phi(q_i)) \qquad (3\text{-}19)$$

also minimizes

$$- \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (r_{i+1} - r_i) \qquad (3\text{-}20)$$

The synthesis problem has been slightly simplified since the Φ_i are more easily calculated than the r_i. However, enumerating G_Q' for a sufficiently large Q may require too much computation. An approximation is now made which will reduce the required computation.

The code function, $\Phi(x)$, will be approximated by a piecewise linear function, $\Phi'(x)$.

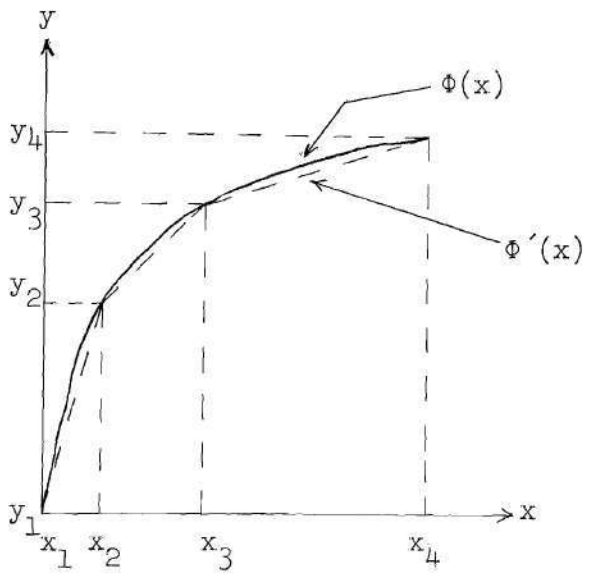


Figure 11. Code Function Approximation

The locations of the breakpoints, x_i and y_i, are yet to be specified. Consider an infinite length input sequence which is just the i^{th} word of the input word set, γ_i, followed by an infinite sequence of zeros. The i^{th} input word, γ_i, will just be coded into δ_i, the i^{th} output word. Since the first word in both the input and output word sets of a CPBA code is a sequence of zeros, the infinite length input sequence of zeros will be transformed into an infinite length output sequence of zeros. The numerical value of the input sequence is obviously just x_i, the numerical value of γ_i, since suffixing zeros to a binary expansion does not change the value of the expansion. Similarly, the numerical value of the output sequence is just y_i, the value of δ_i above. Therefore, any point corresponding to input and output word pairs (x_i, y_i) lies on the code function $\Phi(x)$. Because of this, plus the fact that the values x_i and y_i are closely related to the structure of the input and output word sets, the points (x_i, y_i), for $i = 2, 3, \ldots, N$, will be chosen as the breakpoints of the piecewise linear approximation, $\Phi'(x)$.

The simplification which results from this approximation is the reduction of the infinite sum in expression (3-20) to a finite sum. This is easily demonstrated.

Theorem 5:

Choosing a code which maximizes G_Q

$$G_Q = \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (\Phi'(q_{i+1}) - \Phi'(q_i)) \qquad (3\text{-}21)$$

for arbitrarily large Q is equivalent to choosing the code which maximizes

$$G = \sum_{i=1}^{N} (F_{i+1} - F_i) \log_2 \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{3-22}$$

where

$F_i = F(x_i)$

x_i = numerical value of i^{th} input code word

y_i = numerical value of i^{th} output code word

$i = 1, 2, \ldots, N$

and

$x_{N+1} = y_{N+1} = 1$

The theorem is proved by considering each segment of the piecewise linear approximation, $\Phi'(x)$, separately. It is shown that increasing Q in the expression for G_Q merely adds a constant which can be neglected in the process of maximizing G_Q.

Proof:

The maximum length of a word in the input set of a CPBA code of size N is N - 1, by Theorem 3. Therefore, by Corollary 1

$$x_{i+1} - x_i \geq 2^{-(N-1)} \tag{3-23}$$

$$i = 1, 2, \ldots, N$$

If Q = N - 1, each piecewise linear segment appears as shown in Figure 12. Because of Equation (3-23), it is obvious that $j \geq 1$ in Figure 12, i.e., the q_i may be more closely spaced than the x_i but never less closely spaced.

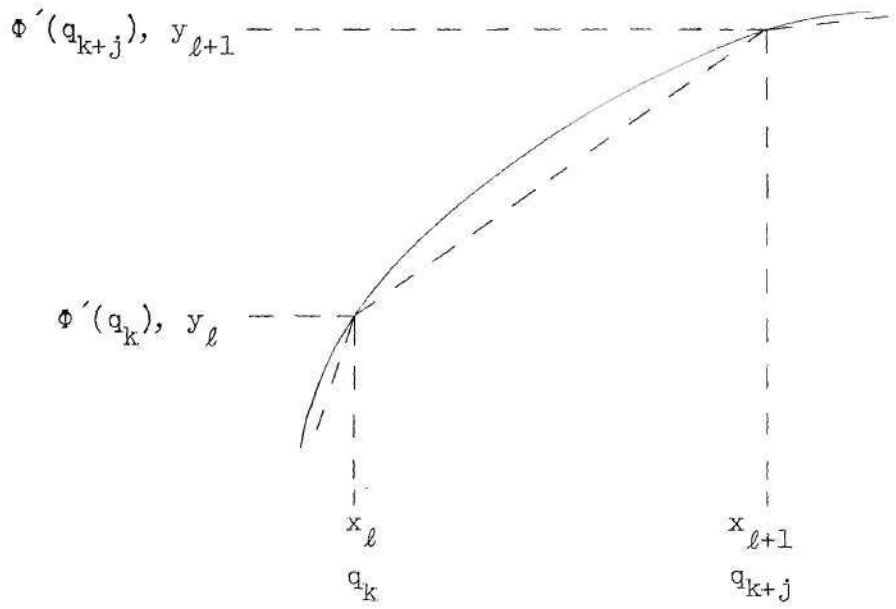


Figure 12. Typical Piecewise Linear Segment

The elements in the summation G_{N-1}, Equation (3-21), which correspond to the segment in Figure 12 are

$$\sum_{i=k}^{k+j-1} (F(q_{i+1}) - F(q_i)) \log_2 (\Phi'(q_{i+1}) - \Phi'(q_i)) \qquad (3\text{-}24)$$

If Q is increased from N - 1 to N, each term in expression (3-24) is replaced by two terms. Let q_i' denote the numerical value of the i^{th} input sequence of length N. (The q_i will still denote the numerical value of i^{th} input sequence of length N - 1.) A typical term of the form

$$(F(q_{i+1}) - F(q_i)) \log_2 (\Phi'(q_{i+1}) - \Phi'(q_i)) \qquad (3\text{-}25)$$

from expression (3-24) will be replaced by the sum of two terms

$$(F(q'_{2i+1}) - F(q'_{2i}))\log_2(\Phi'(q'_{2i+1}) - \Phi'(q'_{2i})) \qquad (3\text{-}26)$$
$$+ (F(q'_{2i}) - F(q'_{2i-1}))\log_2(\Phi'(q'_{2i}) - \Phi'(q'_{2i-1}))$$

The corresponding portion of the input axis is shown in Figure 13.

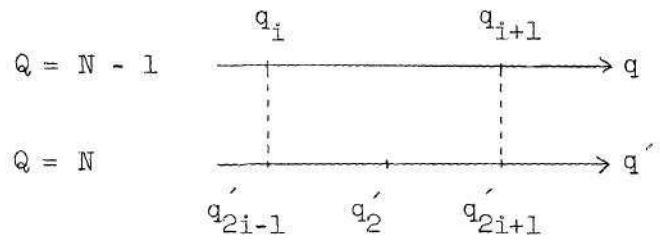


Figure 13. The Input Axis

Recall that this interval (q_i, q_{i+1}) lies on a linear segment of $\Phi'(q)$. Therefore

$$\Phi'(q'_{2i+1}) - \Phi'(q'_{2i}) = \Phi'(q'_{2i}) - \Phi'(q'_{2i-1}) \qquad (3\text{-}27)$$
$$= 1/2(\Phi'(q_{i+1}) - \Phi'(q_i))$$

Hence the two terms in expression (3-26) can be rewritten as

$$(F(q'_{2i+1}) - F(q'_{2i}))\log_2(1/2(\Phi'(q_{i+1}) - \Phi'(q_i))) \qquad (3\text{-}28)$$
$$+ (F(q'_{2i}) - F(q'_{2i-1}))\log_2(1/2(\Phi'(q_{i+1}) - \Phi'(q_i)))$$
$$= (F(q'_{2i+1}) - F(q'_{2i}) + F(q'_{2i}) - F(q'_{2i-1}))(-1 + \log_2(\Phi'(q_{i+1}) - \Phi'(q_i)))$$

$$= - (F(q_{i+1}) - F(q_i)) + (F(q_{i+1}) - F(q_i)) \log_2 (\Phi'(q_{i+1}) - \Phi'(q_i))$$

Since the same argument holds for each linear segment of $\Phi'(q)$

$$G_N = G_{N-1} - \sum_{i=1}^{2^{N-1}} F(q_{i+1}) - F(q_i) \tag{3-29}$$

$$= G_{N-1} - 1$$

It can be shown, similarly, that

$$G_{N+i} = G_{N-1} - (i + 1) \tag{3-30}$$

Therefore, any code which maximizes G_{N-1} maximizes G_Q for arbitrarily large Q.

Consider again the segment $(x_\ell, x_{\ell+1})$ shown in Figure 12. Since $Q = N - 1$

$$j = \frac{x_{\ell+1} - x_\ell}{2^{-(N-1)}} \tag{3-31}$$

By linearity

$$\Phi'(q_{i+1}) - \Phi'(q_i) = \frac{y_{\ell+1} - y_\ell}{j} \tag{3-32}$$

$$= 2^{N-1} \frac{y_{\ell+1} - y_\ell}{x_{\ell+1} - x_\ell}$$

Therefore expression (3-24) can be reduced to

$$\sum_{i=k}^{k+j-1} [F(q_{i+1}) - F(q_i)] \log_2 [\Phi'(q_{i+1}) - \Phi'(q_i)] \qquad (3\text{-}33)$$

$$= \sum_{i=k}^{k+j-1} [F(q_{i+1}) - F(q_i)] \log_2 \left[2^{N-1} \frac{y_{\ell+1} - y_\ell}{x_{\ell+1} - x_\ell} \right]$$

$$= [F(x_{\ell+1}) - F(x_\ell)] \log_2 \left[\frac{y_{\ell+1} - y_\ell}{x_{\ell+1} - x_\ell} \right] + [F(x_{\ell+1}) - F(x_\ell)] \cdot [N - 1]$$

Therefore

$$G_{N-1} = \sum_{i=1}^{N} [F_{i+1} - F_i] \log_2 \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + N - 1 \qquad (3\text{-}34)$$

Since the constant, N-1, does not affect the maximization process, it will be neglected, leaving for the cost function

$$G = \sum_{i=1}^{N} (F_{i+1} - F_i) \log_2 \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \qquad (3\text{-}35)$$

Q. E. D.

Two observations can be made at this point. First, x_1 and y_1 will always be zero in CPBA word sets. By Theorem 4, the possible values of interval lengths such as $x_{i+1} - x_i$ or $y_{i+1} - y_i$ are positive integral powers of 2^{1-N}. Therefore, the sets $\{x_i\}$ and $\{y_i\}$ are contained in the set X_N.

$$X_N = \{k2^{1-N} : k = 0, 1, 2, \ldots, 2^{N-1} - 1\} \qquad (3\text{-}36)$$

Consequently, maximization of G requires consideration of no more than 2^{N-1} possible values for each x_i and y_i.

Second, a Class C code generated by maximizing G is not in general optimum because

1) the code generated will be alphabetic, and

2) the cost function, G, is based on a piecewise linear approximation to the actual code function.

It is shown in this chapter that limitations 1) and 2) do not significantly restrict the efficiency of the synthesized code.

Since the cost function G is nonlinear and separable, it is most practically maximized through the application of dynamic programming. This application is treated in the next section.

Synthesis Procedure

A synthesis procedure which requires less computation than enumeration is now derived by applying dynamic programming to the cost function, G, found in the preceding section. G is repeated below.

$$G = \sum_{i=1}^{N} (F_{i+1} - F_i) \log_2 \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{3-37}$$

The problem is to find the sets $\{y_i\}$ and $\{x_i\}$ which maximize G. Define

$$L_k (x_k, y_k) = \max_{\overline{x_{k-1}}, \overline{y_{k-1}}} \sum_{i=1}^{k-1} (F_{i+1} - F_i) \log_2 \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{3-38}$$

where

$$\overline{x_i} = \{x_1, x_2, \cdots, x_{i-1}, x_i\}$$

and

$$\overline{y_i} = \{y_1, y_2, \cdots, y_{i-1}, y_i\}$$

L_k is just the maximum value of the $(k-1)^{st}$ partial sum of G.

Then

$$L_k(x_k, y_k) = \max_{\overline{x_{k-1}}, \overline{y_{k-1}}} \left[(F_k - F_{k-1}) \log_2 \frac{y_k - y_{k-1}}{x_k - x_{k-1}} + \sum_{i=1}^{k-2} (F_{i+1} - F_i) \log_2 \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right] \tag{3-39}$$

$$= \max_{x_{k-1}, y_{k-1}} \left[(F_k - F_{k-1}) \log_2 \frac{y_k - y_{k-1}}{x_k - x_{k-1}} + \max_{\overline{x_{k-2}}, \overline{y_{k-2}}} \sum_{i=1}^{k-2} (F_{i+1} - F_i) \log_2 \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right]$$

$$= \max_{x_{k-1}, y_{k-1}} [(F_k - F_{k-1}) \log_2 \frac{y_k - y_{k-1}}{x_k - x_{k-1}} + L_{k-1}(x_{k-1}, y_{k-1})]$$

Thus a recursion relation exists between the L_k. This relation reduces the problem of maximizing a function of 2N - 2 variables to the problem of N - 1 maximizations of a function of two variables.

To initiate the maximization procedure, $L_2(x_2, y_2)$ is assigned the value

$$L_2(x_2, y_2) = (F_2 - F_1) \log_2 \frac{y_2 - y_1}{x_2 - x_1}$$

$$= F_2 \log_2 \frac{y_2}{x_2}$$

for all values of x_2 and y_2 which satisfy the CPBA constraints of Theorem 4

$$x_{k+1} - x_k = 2^{m_k+1-N} \tag{3-40}$$

and

$$x_{k+1} - x_k \leq 2^{-\hat{\ell}(x_k)} \tag{3-41}$$

$$k = 1$$

and similarly for y_k. The values of F_1, x_1, and y_1 will all be zero for CPBA word sets. For x_2 and y_2 which do not satisfy constraints (3-40) and (3-41), $L_2(x_2, y_2)$ is assigned a large negative value. Subsequent steps in the maximization will eliminate these values of x_2 and y_2.

Using Equation (3-38), $L_3(x_3, y_3)$ is calculated for all allowed values of x_3 and y_3. The allowed values of x_k or y_k are all but the first $k - 1$ points in the set X_N, Equation (3-36). For each allowed x_3 and y_3, only those values of x_2 and y_2 which satisfy Equations (3-40) and (3-41) are considered. The values of x_2 and y_2 which yield a maximum for a given x_3 and y_3 are denoted by $\hat{x}_2(x_3, y_3)$ and $\hat{y}_2(x_3, y_3)$.

This procedure is continued sequentially, that is, $L_k(x_k, y_k)$ is calculated for all allowed values of x_k and y_k from the values of $L_{k-1}(x_{k-1}, y_{k-1})$ making sure that the x_{k-1} and y_{k-1} satisfy Equations (3-40)

and (3-41). At each step the values of $\hat{x}_{k-1}(x_k, y_k)$ and $\hat{y}_{k-1}(x_k, y_k)$ are recorded. After N - 1 maximizations, the synthesized sets, $\{x_i\}$ and $\{y_i\}$, can be determined. First x_N and y_N are chosen.

$$x_N = \hat{x}_N(1, 1) \tag{3-42}$$

and

$$y_N = \hat{y}_N(1, 1) \tag{3-43}$$

Then the other x_k and y_k can be determined sequentially from Equations (3-44) and (3-45).

$$x_k = \hat{x}_k(x_{k+1}, y_{k+1}) \tag{3-44}$$

and

$$y_k = \hat{y}_k(x_{k+1}, y_{k+1}) \tag{3-45}$$

for $k = N - 1, N - 2, N - 3, \ldots, 2$

The i^{th} input word is then just the first ℓ_i digit in the binary expansion of x_i, and similarly for the output words. The $\{\ell_i\}$ are determined from the $\{x_i\}$ by Theorem 2 which states

$$\ell_i = 2^{-(x_{i+1}-x_i)} \tag{3-46}$$

The $\{m_i\}$ are similarly determined from the $\{y_i\}$. Given both word sets, the code efficiency can be calculated.

To summarize, an ergodic, binary random process is assumed to represent a physical source. The probability of a zero being emitted by

the source is assumed known. An expression for the limit of the expected value of the compression ratio is derived. This expression, which is an infinite sum, is reduced to a finite sum, G, by approximating the code function with a piecewise linear function. Since G is separable, it can be maximized by the application of dynamic programming. Constraints are included in the maximization to insure that the resulting word sets are CPBA. The efficiencies of codes synthesized by this method are calculated in the next section for various sizes and various source statistics.

Basic Class C Results

Since functional synthesis produces alphabetic codes and assumes a piecewise linear approximation to the code function, it is important to compare the efficiencies of functionally synthesized codes with those of the optimum codes as determined by enumeration. As will be shown in the next section, the computation required for enumeration is actually less than that required for functional synthesis for small sized codes. As code size increases, the computation required for enumeration quickly exceeds that required for functional synthesis.

The efficiencies of optimum and functionally synthesized codes of sizes three through seven for zero memory binary sources having entropies of .1, .5, and .9 are presented in Figures 14 and 15. Recall that for binary output alphabets, the efficiency is just the number of bits per output symbol.

As can be seen in Figure 14 the efficiencies of the optimum and functionally synthesized codes are plotted as the same curve for entropies of .1 and .5. The curves are identical to five significant figures. For

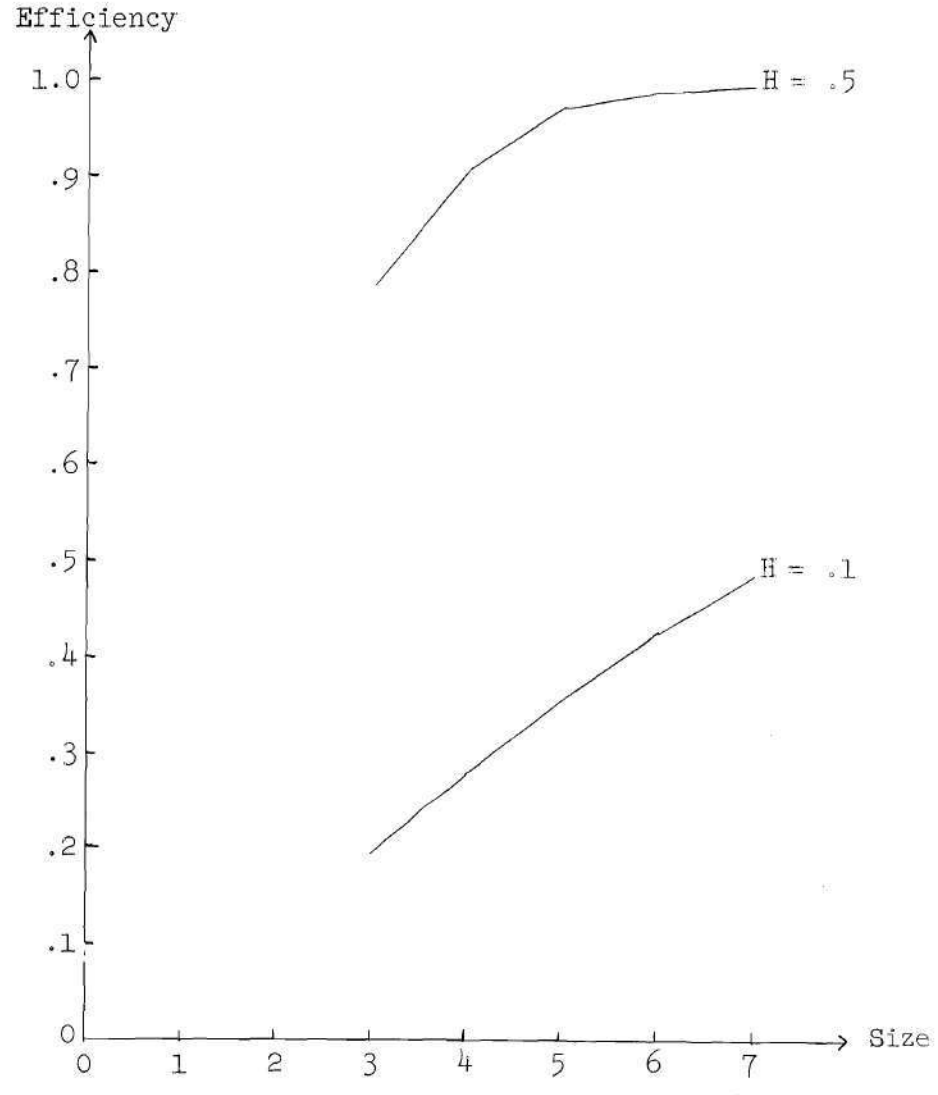


Figure 14. Optimum and Functional Class C Code Efficiencies

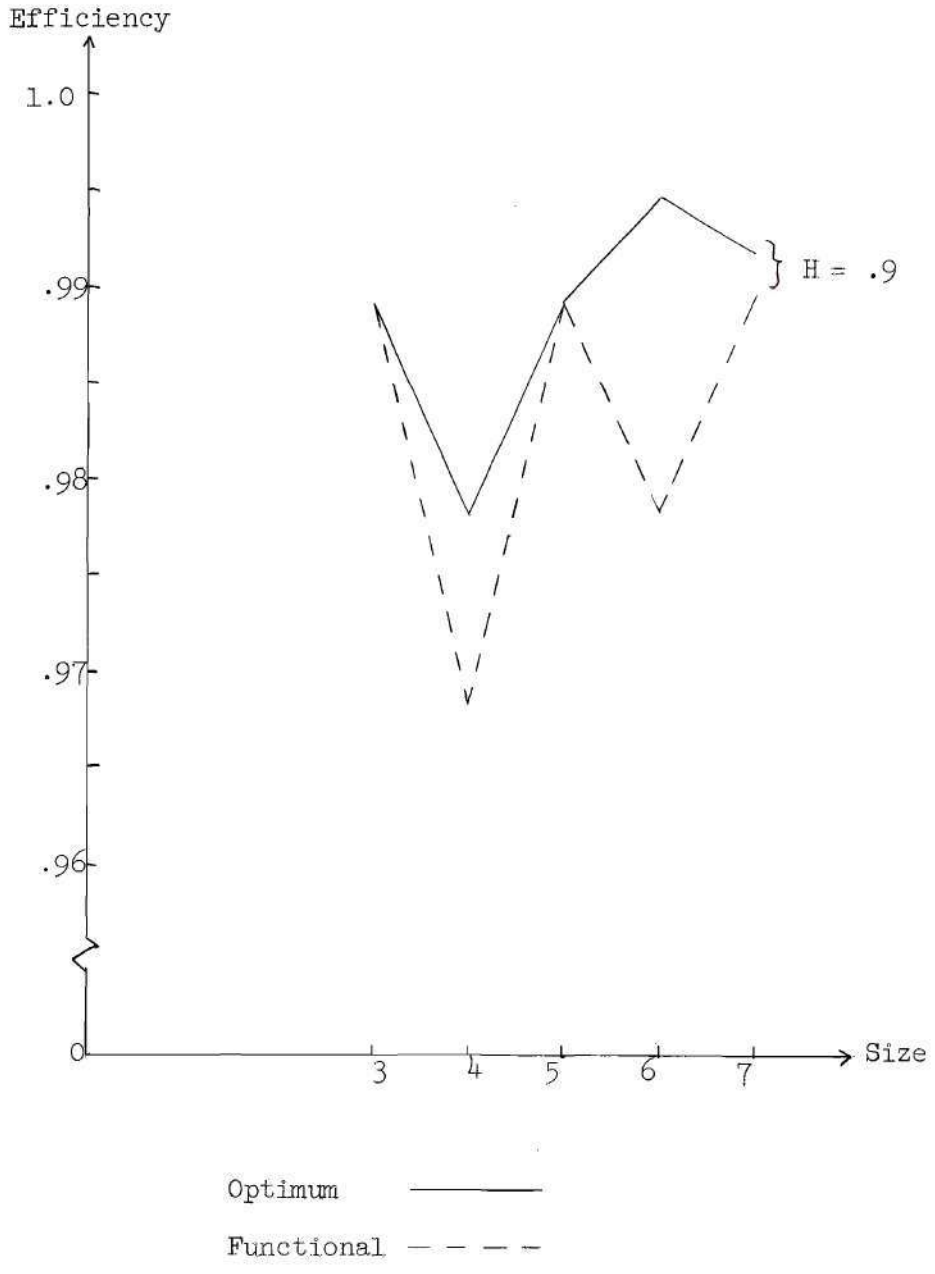


Figure 15. Optimum and Functional Class C Code Efficiencies

entropy of .9 as shown in Figure 15, the maximum deviation between the two curves is about 1.6 percent.

Hence, it appears that for a wide range of sizes and statistics the difference in efficiencies of the optimum and the functionally generated codes is negligible.

Computational Requirements

Bounds on the computational requirements of enumeration and functional synthesis are now determined. Consider first the case of enumeration.

A binary word set of size N is uniquely specified by the locations of the words modified in each of the N - 2 suffixing operations. Two possible locations exist for the first operation, three for the second, etc., and N - 1 for the $(N - 2)^{nd}$. Therefore, the number of possible unique CPBA word sets of size N is $(N - 1)!$. If the source is zero memory and the output letters are equal cost the input word set can be enumerated and the output set can be determined by Huffman's procedure. Since the computation required for Huffman's procedure is approximately proportional to N, the total computation required is proportional to $N!$.

In the more general case of a Markov source or unequal cost output letters, both word sets must be enumerated and the computation required is proportional to $[(N - 1)!]^2$.

Consider now the case of functional synthesis. In either the simple case or the more complex cases mentioned above, the number of possible values for the members of the $\{x_i\}$ set is $1 + 2^{N-1} - N$ or approximately 2^{N-1}. Similarly for the $\{y_i\}$ set. Therefore, an upper bound on the com-

putation required in functional synthesis is $K\ 4^N$, where K is a constant.

Obviously for code sizes much larger than 4, the functional procedure will be faster than enumeration.

Functional synthesis has now been defined, shown to be near optimum, and shown to be computationally advantageous. In the next chapter, functional synthesis is extended to some important special cases.

CHAPTER IV

GENERALIZATIONS OF FUNCTIONAL SYNTHESIS

The basic functional synthesis technique can easily be extended to the special cases of Markov sources, unequal cost output letters, constrained maximum word lengths, Class A codes, or Class B codes.

Markov Sources

In the preceding chapter no specific assumption is made concerning the statistical nature of the source. All that is assumed is that the probability function, $F(q_i)$, can be calculated. This condition is satisfied by Markov sources as well as zero memory sources. The information necessary to calculate $F(q_i)$ for a Markov source is the set of transition probabilities and the initial probabilities.

Assume the Markov source is binary and has initial probabilities

$$I_0 = \Pr \text{ (initial 0)}$$

and

$$I_1 = \Pr \text{ (initial 1)}$$

Transition probabilities will be denoted by

$$S_{ij} = \Pr (j|i) \qquad i, j = 0, 1$$

Therefore the probability that a sequence begins with a zero is I_0. The

probability that a sequence begins with 01 is $I_0 S_{01}$. To illustrate the calculation of $F(x_i)$ consider $F(5/8)$.

$$F(5/8) = \Pr \text{ (all sequences beginning with 0)}$$
$$+ \Pr \text{ (all sequences beginning with 1 0 0)}$$
$$= I_0 + I_1 S_{10} S_{00}$$

Calculation of the values of $F(q_i)$ for use in the maximization of G, thus requires both an initial probability set and a transition probability set. It is assumed that the transition probabilities are specified but that the initial probabilities are unknown. Since the initial probabilities are independent of the transition probabilities, they can be specified arbitrarily. In the following examples the initial probabilities will be chosen equal to the steady state probabilities, S_i. Examples indicate that this minimizes the degradation in efficiency due to approximating the code function. The steady state and initial probabilities are, therefore, both solutions to Equations (4-1) and (4-2). See reference (12).

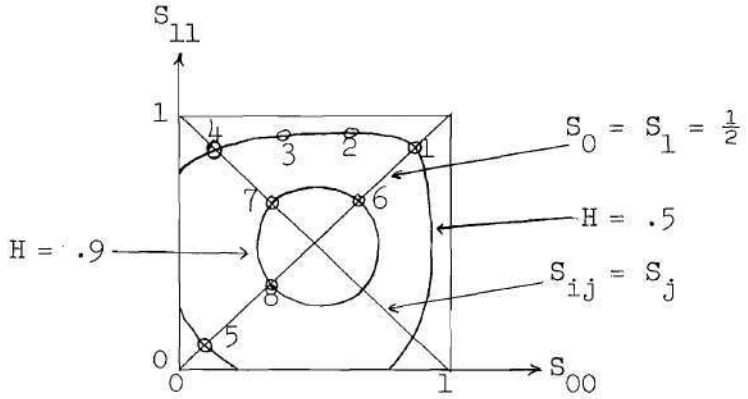
$$S_0 S_{00} + S_1 S_{10} = S_0 \qquad (4\text{-}1)$$

$$S_0 + S_1 = 1 \qquad (4\text{-}2)$$

The entropy of a Markov source is given by

$$H = - \sum_{j=0}^{1} \sum_{i=0}^{1} S_i \, S_{ij} \log_2 S_{ij} \qquad (4\text{-}3)$$

Obviously there are an infinite number of sets of S_{ij} which yield the same entropy, H. The values of S_{ij} chosen for examples lie on the contours of constant entropy in the S_{00}, S_{11} plane. The points chosen are shown in Figure 16.



$$S_{ij} = Pr(j|i) \qquad S_i = Pr(i)$$

$$H = - \sum_{i=1}^{2} \sum_{j=1}^{2} S_i S_{ij} \log_2 S_{ij}$$

Figure 16. Constant Entropy Contours

The contours chosen correspond to entropies of 0.5 and 0.9. As indicated in Figure 16, the line $S_{00} = S_{11}$ corresponds to sources for which $S_0 = S_1 = 1/2$, i.e., the zero and one are equally likely. However, in the lower left corner long runs of alternation between zero and one are likely; in the upper right corner long runs of ones or long runs of zeros are likely.

The other diagonal, i.e.,

$$S_{11} = 1 - S_{00} \tag{4-4}$$

corresponds to zero memory sources. This is easily shown by substituting Equation (4-4) into Equation (4-1).

The efficiencies of functionally synthesized Class C codes of size seven are tabulated in Table 1 for various points on the constant entropy contours.

Table 1. Code Efficiencies for Markov Source

Point	H	S_{00}	S_{11}	Efficiency
1	.5	.89	.89	.759
2	.5	.65	.92	.817
3	.5	.35	.9091	.936
4	.5	.11	.89	.993
5	.5	.11	.11	.728
6	.9	.684	.684	.951
7	.9	.316	.684	.989
8	.9	.316	.316	.948

Two trends should be noted in these results. First, along either constant entropy contour the efficiencies at points close to the zero memory diagonal are greater than at points close to the equiprobable symbol diagonal. Let E_i denote the efficiency at point i. E_4 is greater than E_1 or E_5; E_7 is greater than E_6 or E_8.

Second, along the diagonal corresponding to equiprobable symbols, $S_{11} = S_{00}$, the efficiencies at points close to the zero memory diagonal are greater than the efficiencies at more extreme points. E_6 or E_8 is

greater than E_1 or E_5.

These two tendencies will be noted again in the next chapter with regard to facsimile encoding.

The codes synthesized for Markov sources are intuitively satisfying. Consider the codes synthesized for the probabilities at points 1 and 5 on Figure 16. These codes are shown in Figures 17 and 18.

0 0 0	→	0
0 0 1	→	1 0 0 0
0 1	→	1 0 0 1
1 0	→	1 0 1 0
1 1 0	→	1 0 1 1 0
1 1 1 0	→	1 0 1 1 1
1 1 1 1	→	1 1

Figure 17. Code for $S_{00} = S_{11} = .89$

0	→	0
1 0 0	→	1 0 0 0
1 0 1 0 0	→	1 0 0 1
1 0 1 0 1 0	→	1 0 1
1 0 1 0 1 1	→	1 1 0 0
1 0 1 1	→	1 1 0 1
1 1	→	1 1 1

Figure 18. Code for $S_{00} = S_{11} = .11$

Note that in the code in Figure 17 the most frequent patterns, runs or zeros or ones, are coded into the shortest output code words. In the code of Figure 18, the expected pattern, a run of ones and zeros appears

in the fourth input word; it is coded into an output word of length three which is shorter than most of the other output words.

The extension of functional synthesis to the case of m^{th} order Markov sources, $m > 1$, is straightforward. It, however, adds no new insights and is not proved here.

The extension of functional synthesis to the case of Markov sources places it in strong contrast with the optimal Class B synthesis procedure presented in Chapter II and the available synthesis procedures mentioned in Chapter I, all of which are constrained to zero memory sources.

Unequal Cost Output Letters

The derivation of the functional synthesis cost function is based on Theorem 2 which implies

$$- \log_2 (y_{i+1} - y_i) = m_i \qquad (4\text{-}5)$$

$$i = 1, 2, \ldots, N$$

where $\{y_i\}$ and $\{m_i\}$ are the numerical values and lengths, respectively, of a CPBA output word set. It is assumed in Theorem 4 that the symbols are equal cost, i.e., equal duration, and m_i is just the number of symbols in the i^{th} output word. It is now shown that functional synthesis can be extended to the case of arbitrary output letter durations.

Define the durations of the output zero and output one to be

$$C_0 = \text{duration of output zero}$$

and

$$C_1 = \text{duration of output one}$$

Further, define

$$A_0 = \frac{2^{-C_0}}{2^{-C_0} + 2^{-C_1}} \tag{4-6}$$

and

$$A_1 = \frac{2^{-C_1}}{2^{-C_0} + 2^{-C_1}} \tag{4-7}$$

Since

$$A_1 + A_0 = 1$$

the output axis, i.e., the y axis, coordinates can be non-linearly spaced in the ratio $A_0:A_1$. For example, the coordinates of sequences of length two would be as shown in Figure 19.

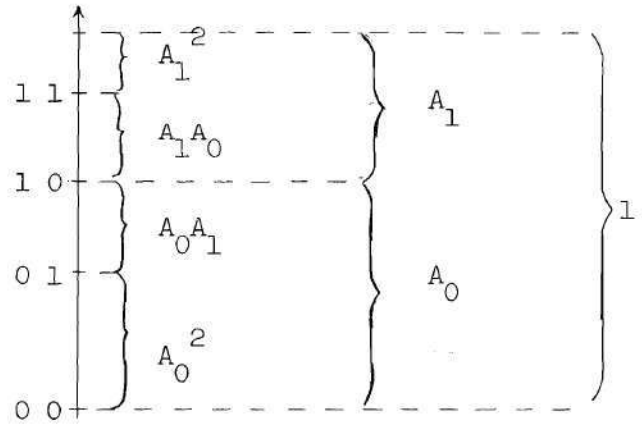


Figure 19. Output Axis Spacing for Unequal Cost Output Letters

Finally, define

$$m_i = \text{length of } i^{th} \text{ output word}$$

$$= m_{i0}C_0 + m_{i1}C_1$$

where

$$m_{i0} = \text{number of zeros in } i^{th} \text{ output word}$$

and

$$m_{i1} = \text{number of ones in } i^{th} \text{ output word}$$

Let w_i be the value along the non-linearly spaced output axis of the i^{th} output word. For example, if the third output word is 11, then

$$w_3 = A_1A_0 + A_0$$

It is now shown, using an inductive proof, that a relation similar to Equation (4-5) exists for unequal cost output letters. The functional cost function and the synthesis procedure based on it are therefore valid for the case of unequal cost output letters.

Theorem 6:

$$- \log_2 (w_{i+1} - w_i) = m_i \qquad (4\text{-}8)$$

$$i = 1, 2, \ldots, N$$

where

$$w_{N+1} = 1$$

Proof:

Assume N = 2. The two code words in the output word set will be

$$\delta_1 = 0 \qquad \delta_2 = 1$$

By definition

$$w_1 = 0 \qquad \text{and} \qquad w_2 = A_0$$

Therefore,

$$- \log_2 (w_2 - w_1) = - \log_2 A_0$$

$$= - \log_2 \frac{2^{-C_0}}{2^{-C_0} + 2^{-C_1}}$$

$$= C_0 + \log_2 (2^{-C_0} + 2^{-C_1})$$

It will be assumed that C_0 and C_1 have been normalized keeping C_0/C_1 constant so that

$$2^{-C_0} + 2^{-C_1} = 1 \tag{4-9}$$

and

$$- \log_2 (w_2 - w_1) = C_0 + \log_2 1$$

$$= C_0$$

Similarly

$$- \log_2 (w_3 - w_2) = C_1$$

Now assume the theorem is valid for any word set of size N

$$N > 2$$

It will be shown that the theorem must be valid for size N + 1, and the

proof will follow by induction. Assume the word set of size N + 1 is formed by modifying the j^{th} word in the word set of size N. The values of the w_i for both word sets would appear as shown in Figure 20.

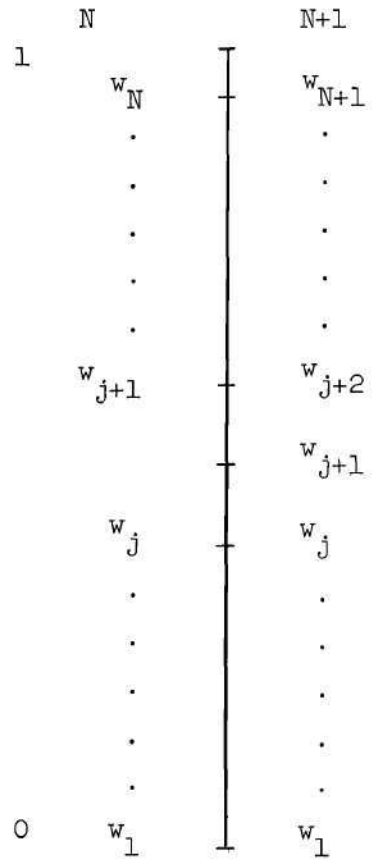


Figure 20. Output Axis Spacing

For the set of size N + 1 it is obvious that

$$w_{j+1} - w_j = A_0 (w_{j+2} - w_j)$$

Therefore

$$- \log_2 (w_{j+1} - w_j) = - \log_2 A_0 - \log_2 (w_{j+2} - w_j)$$

$$= C_0 + \ell_j$$

where

ℓ_j = length of j^{th} word in set of size N

Similarly

$$- \log_2 (w_{j+2} - w_{j+1}) = C_1 + \ell_j$$

Of course, the lengths of the words in the set of size N + 1 will be unchanged for $i \leq j$ and $i \geq j + 2$. Thus, by induction

$$- \log_2 (w_{i+1} - w_i) = m_i$$

$$i = 1, 2, \ldots, N$$

for any $N \geq 2$, if

$$2^{-C_0} + 2^{-C_1} = 1$$

Q. E. D.

Because of the similarity of Equations (4-8) and (4-5), only a few modifications are required to extend functional synthesis to the case of unequal cost output letters. Theorem 6 implies that Equation (3-15) can be rewritten as

$$E(R) = - \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (d_{i+1} - d_i) \qquad (4\text{-}10)$$

where d_i is the numerical value of the i^{th} output sequence using non-linearly spaced output coordinates. Using a proof similar to that in the Appendix, it can be shown that the limit of the expected compression ratio

is

$$\lim_{Q \to \infty} \frac{E[R]}{Q} = - \lim_{Q \to \infty} \frac{1}{Q} \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (D(q_{i+1}) - D(q_i)) \tag{4-11}$$

Here, $D(q)$ is the value of $\Phi(q)$ for a non-linearly spaced output axis. The function $D(q)$ can be piecewise linearly approximated using the points (x_i , w_i) for the breakpoints. The resulting cost function is

$$G' = \sum_{i=1}^{N} (F_{i+1} - F_i) \log_2 \frac{w_{i+1} - w_i}{x_{i+1} - x_i} \tag{4-12}$$

Because G' is separable, it is easily maximized by an application of dynamic programming. Note that although y_i does not appear in Equation (4-12), it must still be used to apply the constraints of Theorem 4 when maximizing G'. With these few modifications, functional synthesis can be extended to the case of unequal cost output letters.

In Figure 21, the efficiencies of functionally synthesized Class C codes of size seven are plotted for a zero memory source of entropy .5 and for a range of values of letter costs, C_0 and C_1. Note that the highest efficiency occurs when the costs are equal.

Constrained Maximum Word Lengths

As mentioned in Chapter III, the values of x_k and y_k considered in the maximization operations are constrained by Theorem 4, which states

$$x_{k+1} - x_k = 2^{m_k + 1 - N} \tag{4-13}$$

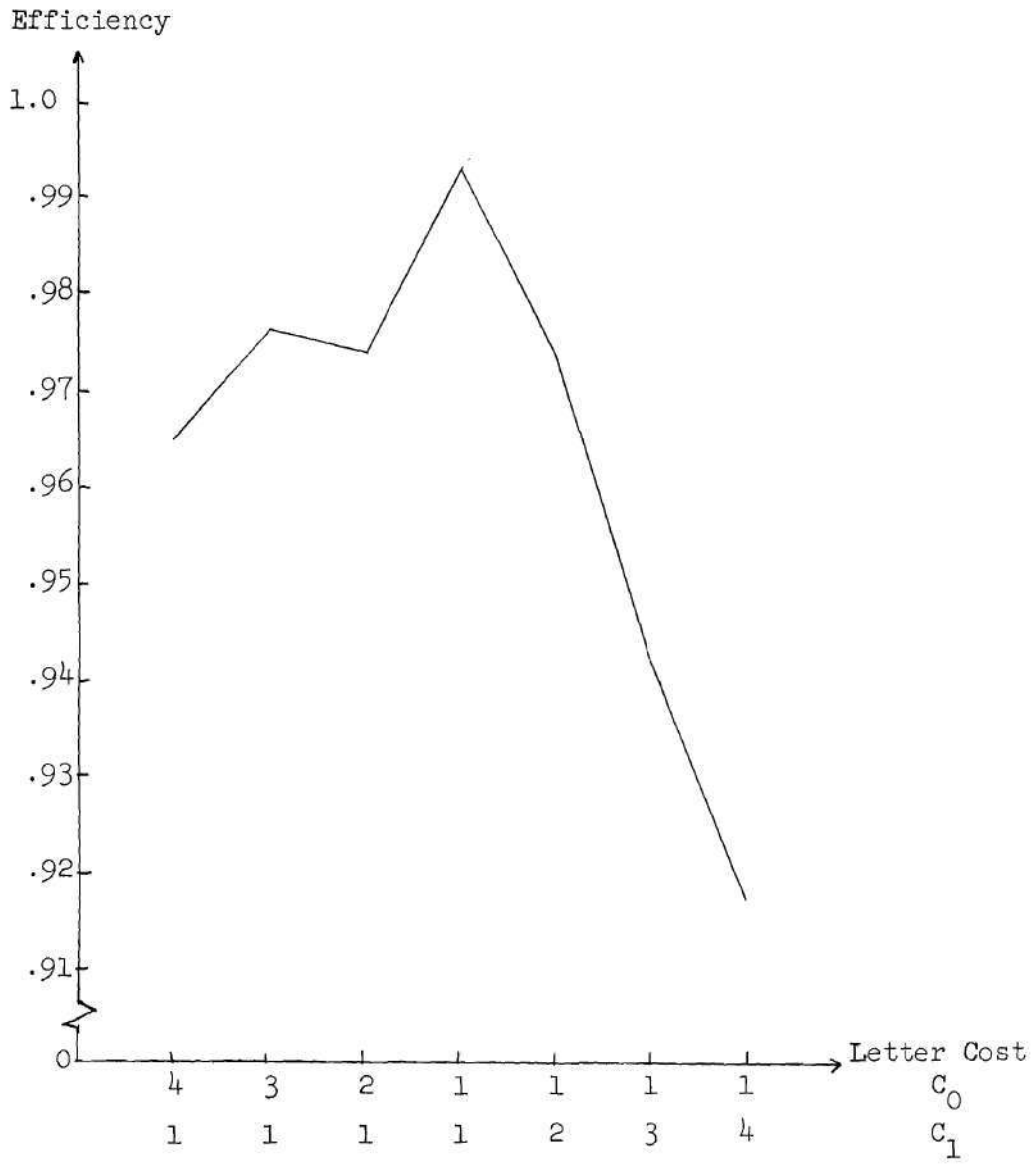


Figure 21. Unequal Cost Letters

and

$$x_{k+1} - x_k \leq 2^{-\hat{\ell}(x_k)} \tag{4-14}$$

Similar constraints apply to the y_i.

If it is desired to synthesize near-optimal codes in which the input word lengths are less than or equal to ℓ_{max} and the output word lengths are less than or equal to m_{max}, then it is necessary only to add to the constraints (4-13) and (4-14), the constraint

$$x_{k+1} - x_k \geq 2^{-\ell_{max}} \tag{4-15}$$

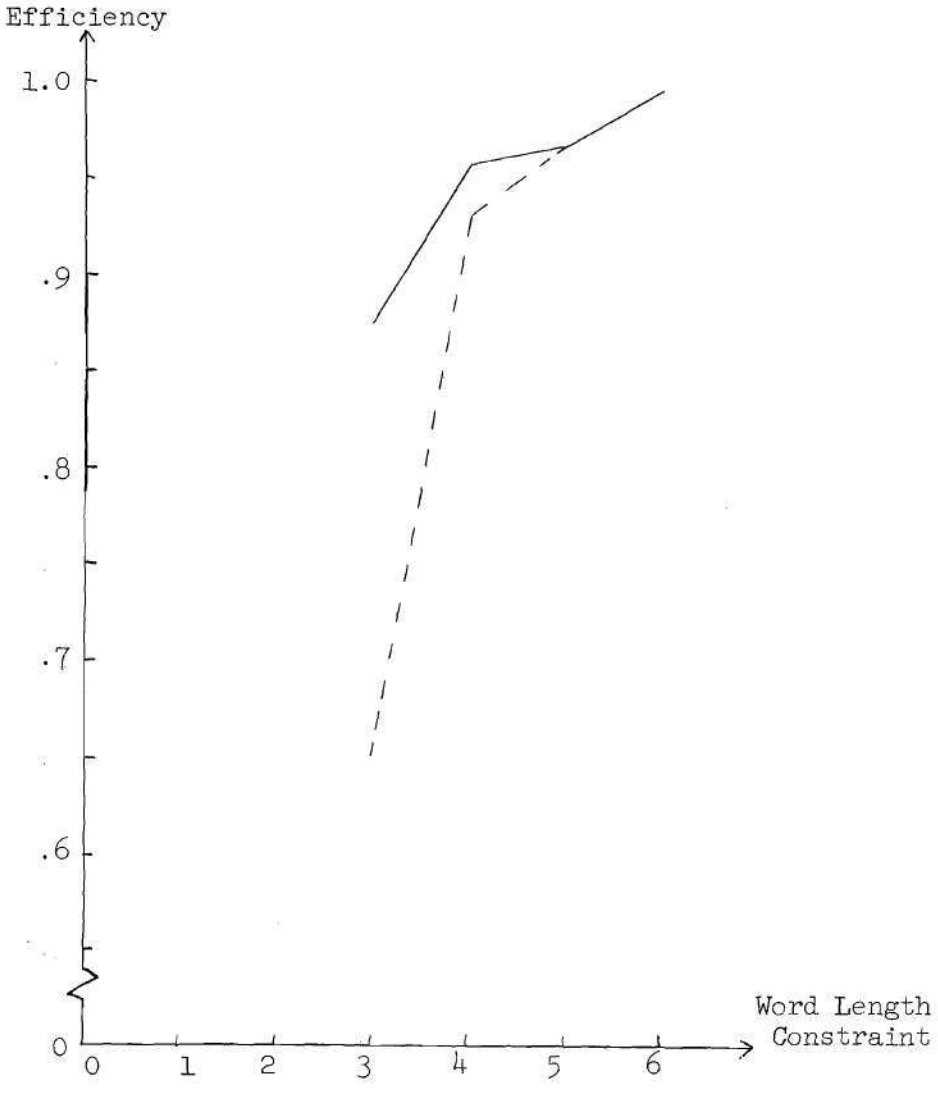
$$k = 1, 2, \ldots, N$$

and similarly for the y_i. This follows from Corollary 1.

Thus in each step of the maximization of the functional cost function, G, possible values of x_{k-1} and y_{k-1} are selected from set X_N, Equation (3-36), if they satisfy the constraints of Theorem 4. From these subsets are removed values of x_{k-1} and y_{k-1} which violate Equation (4-15). The remaining subsets, which can be shown to be non-empty, are considered in the maximization.

Therefore, near-optimum codes can be synthesized which are CPBA and which have maximum input and output word lengths no greater than ℓ_{max} and m_{max}, respectively.

The efficiencies of functionally synthesized Class C codes of size seven, for which one or both word sets have maximum word length constraints, are shown in Figure 22. The source is assumed to be zero memory and to



——— Input Word Constrained,
Output Word Unconstrained

– – – – Both Words Constrained

Figure 22. Constrained Word Lengths

have entropy of .5. As is to be expected, efficiency suffers more when both word sets are constrained, rather than just one.

Restrictions on maximum code word length might result from bounds on equipment complexity or encoding delay. In addition, constraints on maximum word length can greatly reduce computation time for synthesis of a code of a given size.

Class A and B Codes

It is shown in earlier sections that constraints applied to the sets $\{x_i\}$ and $\{y_i\}$ at each step in the maximization of G can result in near-optimum codes having particular properties. The use of constraints from Theorem 4 implies that the synthesized code will be CPBA. The use of constraint (4-15) implies the length of the longest input word in the synthesized code is no greater than ℓ_{max}. It is now shown that constraints can be applied to the sets $\{x_i\}$ or $\{y_i\}$ which will result in Class A or Class B codes.

Assume it is desired to synthesize a CPBA Class A code of size

$$N = 2^K$$

The input word set will be CPBA and contain N words of length K. Assume the set $\{x_i\}$ is constrained so that

$$x_i = (i-1)2^{-K} \tag{4-16}$$

Since Equation (4-16) satisfies the constraints of Theorem 4, the set $\{x_i\}$ is CPBA. Therefore, by Theorem 2

$$\ell_i = K$$

$$i = 1, 2, \ldots, N$$

which is the desired result.

To summarize, at each step in the maximization of G, y_{k-1} is chosen to satisfy the constraints of Theorem 4, and x_{k-1} is set equal to $(k-1)2^{-K}$. The resulting code is CPBA and Class A.

Conversely, the set $\{y_i\}$ can be equally spaced, which results in a Class B code.

The efficiencies of Class A and B codes synthesized in this manner for zero memory sources are shown in Figure 23. Most of these efficiencies are identical within five decimal places to the efficiencies of optimum Class A and B codes shown in Figure 8. Only the efficiencies of Class A codes of size eight differ substantially from the optimum values and in each case by less than three percent.

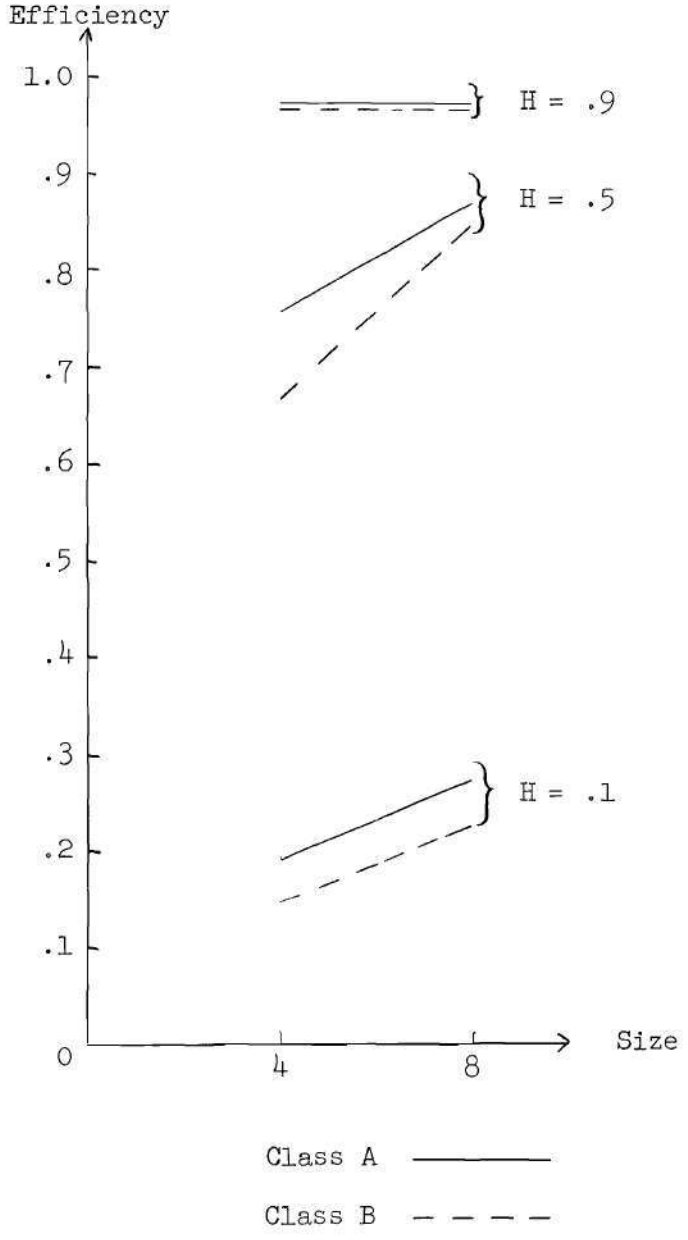


Figure 23. Functional Class A and B Code Efficiencies

CHAPTER V

FACSIMILE ENCODING

Attendant with any application of compression coding to a practical situation are numerous problems, for example, the estimation of unknown or nonstationary source statistics, error control, synchronization, and buffer overflow. Generally, these problems are very difficult to solve, and they depend on the characteristics of the particular source involved. One aspect of the problem of coding facsimile images is considered in this chapter. The effects on compression of various parameters of the images are determined and are compared with the results of Chapter IV.

Facsimile Images

Consider two situations involving the transmission of facsimile information. In the first situation a facsimile transmission link is to be used for a particular class of images, e.g., business letters, blueprints, or newspaper wirephotos. To determine a compression code, the statistical parameters of a small number of representative images of a given class are determined. In the absence of a priori information concerning the physical basis of the image class, the statistical parameters are estimated by some form of relative frequency measurement.

In the second situation, an adaptive compression coder is desired. The input images do not all belong to the same class but the images occur in groups of one class and then another. In this situation the coder must

form a running estimate of the statistical parameters of the input images. Again, some form of relative frequency measurement will be involved.

In this chapter, these two situations are reduced to their simplest extreme, a single image. Compression codes based on the relative frequency measurements of this image are used to encode the image and the compression ratios are tabulated.

For an image which is divided into N grid cells, the unconditional probabilities are estimated by

$$S_i = n_i/N \tag{5-1}$$

where

$i = 0, 1$

n_0 = number of black cells

and

n_1 = number of white cells

If a cell is more than half black, it is counted as black and vice versa. The conditional probabilities are estimated by

$$S_{ij} = n_{ij}/n_i \tag{5-2}$$

where

n_{ij} = number of transitions from i to j (e.g., from black to white)

The images considered are shown in Figure 24. The circles have areas of one-half and one-tenth the area of the square. It is assumed that the disk is black on a white square. The grid size is one-hundredth or one-twentieth the edge length of the square. The images were digitized

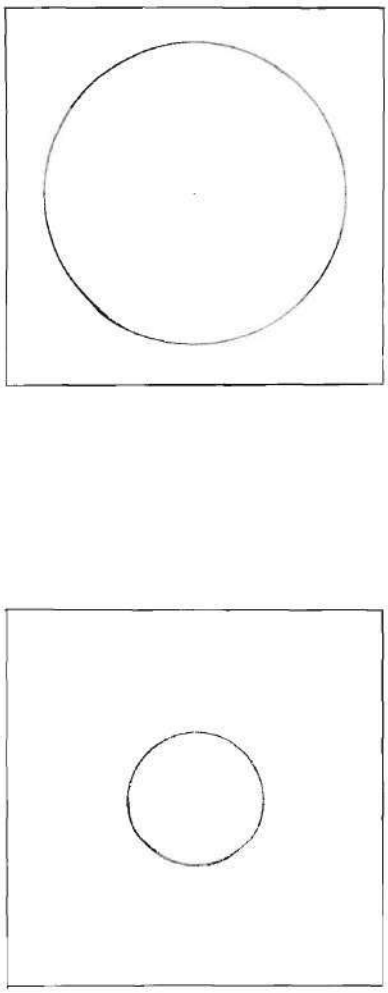


Figure 24. Facsimile Images

by plotting them on graph paper and counting n_i, and n_{ij} by hand. Formulas (5-1) and (5-2) were applied and the estimated probabilities were used for synthesizing Class C codes of size seven. Using these synthesized codes, the images were coded by hand and the compression ratios recorded.

Facsimile Compression

The facsimile experiments in this section were selected to determine the effects on the compression ratio of two types of changes in the image. The first type of change is a change in the grid size of the half white, half black image. Recalling Figure 16 in the section on Markov sources, this is equivalent to moving along the equiprobable symbol diagonal.

The second type of change is reducing the disk size. This corresponds to moving along a constant entropy contour in Figure 16.

In addition, for each image, codes were synthesized from both unconditional and conditional probability estimates.

The results are tabulated in Table 2.

Table 2. Facsimile Results

Experiment	S_0	Order of Approximation	S_{00}	S_{11}	Cell Size	Compression Ratio
1a	.5	0	.5	.5	1/100	1.00
1b	.5	1	.984	.984	1/100	.455
2a	.5	0	.5	.5	1/20	1.00
2b	.5	1	.92	.92	1/20	.290
3a	.1	0	.9	.1	1/100	.438
3b	.1	1	.966	.964	1/100	.347

The compression ratio tabulated in Table 2 is the ratio of the output sequence length to the input sequence length, N. Therefore, the smaller the compression ratio, the more efficient the code.

Three tendencies can be noted in these results.

1) Whenever the approximation order goes from zero to one, i.e., whenever conditional rather than unconditional probabilities are considered, and other parameters remain fixed, the compression ratio improves.

2) In going from experiment 1b to 2b, the unconditional probabilities remain one-half, but the average run lengths are shortened due to the larger grid size. This is equivalent to moving from the upper right to the lower left along the equiprobable symbol diagonal in Figure 16. In agreement with the results of Chapter IV, the compression ratio improves.

3) In going from experiment 1b to 3b, the probability point in the S_{00}, S_{11} plane shifts from (.984, .984) to (.996, .964). This is approximately equivalent to travelling along a contour of constant entropy from the upper right corner, in a clockwise direction. Again, in agreement with the results of Chapter IV, the compression ratio improves.

These experimental results plus the results of Chapter IV indicate that for a code of specified size the greatest facsimile compression will be obtained when conditional statistics are used and the average lengths of white runs and black runs are short.

CHAPTER VI

CONCLUSIONS

The objective of this research has been to determine synthesis procedures for noiseless compression codes. This area was chosen for investigation because of the increasing importance of digital data transmission over channels having low error rate and because of the lack of significant work on Class B and Class C codes. It was hoped that efficient synthesis procedures could be found which were dependent on combinatorial algorithms or on standard optimization tools such as calculus of variations, gradient methods, linear programming, and dynamic programming.

Consideration of the combinatorial approach to Class C code synthesis led to the optimum Class B algorithm. The method of attack and the proof by induction were influenced mainly by Huffman's work on Class A codes (2).

The functional concept was investigated next since it appeared to offer a separable cost function and to cover several special cases. The way in which use was made of the functional concept was influenced mainly by the work of Billingsley (5) and that of Laemmel (11).

Research Results

The two major results of the research are:

1) An optimum combinatorial Class B synthesis procedure has been

found. It is valid for any size input or output alphabet. It is constrained to the cases of zero memory sources and equal cost output letters.

2) A near-optimum functional synthesis procedure has been found. It is valid for Markov sources, output letters of arbitrary cost, constrained word lengths, and Classes A, B, or C. It is constrained to binary output alphabets.

These synthesis procedures are a valuable contribution because of their exploitation of non-uniform length input word sets and because of their flexibility. Also, the rigorous derivation of the cost function, G, in Equation (2-14), is significant and has not appeared in the literature previously.

$$G = \frac{\sum_{i=1}^{N} p_i \, m_i}{\sum_{i=1}^{N} p_i \, \ell_i} \qquad (2\text{-}14)$$

Suggestions for Further Study

The first and most obvious question for further study is the combinatorial synthesis of Class C codes. As pointed out in Chapter II, there cannot be a sequential combinatorial procedure for an optimum Class C code. However, the possibility of a non-sequential algorithm for optimum Class C should be investigated further. Also, it should be possible to find near-optimal sequential algorithms along with efficiency bounds.

Another class of codes which might be profitably studied is multi-component codes (7). The input and output word sets change according to

symbols previously encoded. Synthesis of this type code is obviously more complicated than synthesis of Classes A, B, or C codes.

A last suggestion for further study is a system in which a compression coder is followed by an error-correcting coder. It is usually assumed that the input of an error-correcting coder consists of equi-probable, independent symbols; this is not generally true of the output of a compression coder. Rather than synthesizing the compression coder and the error-correcting coder independently, it may be possible to jointly synthesize them and realize a given error rate bound with less equipment.

APPENDIX

LIMIT OF FUNCTIONAL COMPRESSION RATIO

The limit which was merely stated in Equation (3-18) will now be proven.

Theorem 7:

$$\lim_{Q \to \infty} E(R) / Q = - \lim_{Q \to \infty} 1/Q \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (r_{i+1} - r_i)$$

$$= - \lim_{Q \to \infty} 1/Q \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (\Phi_{i+1} - \Phi_i)$$

where

Q = input sequence length

R = output sequence length

q_i = numerical value of i^{th} input sequence

r_i = numerical value of i^{th} output sequence

$F(q)$ = Pr(input sequence numerical value is less than q)

$\Phi_i = \Phi(q_i)$

= value of code function at q_i

Basically the proof of Theorem 7 depends on finding bounds on the ratio of $(\Phi_{i+1} - \Phi_i)$ to $(r_{i+1} - r_i)$. To simplify the proof, some lemmas are first proved.

Recall that in finite length encoding, the input sequence is encoded until the uncoded remainder is shorter than the longest word in the

input set. This remainder is then transmitted in uncoded form. All of the sequence preceding the uncoded remainder is called the prefix. A hat, "^", will be used to denote sequences or numerical values in which the digits corresponding to the remainder have been deleted. Specifically,

π_i = i^{th} input sequence

$\hat{\pi}_i$ = i^{th} input sequence prefix

σ_i = i^{th} output sequence

$\hat{\sigma}_i$ = i^{th} output sequence prefix

$\hat{q}_i$ = numerical value of i^{th} input prefix

$\hat{r}_i$ = numerical value of i^{th} output prefix

$\hat{Q}_i$ = length of i^{th} input sequence prefix

R_i = length of i^{th} output sequence

$\hat{R}_i$ = length of i^{th} output sequence prefix

Define

$$q_{2^Q+1} = \Phi_{2^Q+1} = r_{2^Q+1} = 1.00$$

Let

Z_i = length of remainder of π_i

Lemma 1:

If $Z_i = 0$, then $\Phi_i = r_i$, $i = 1, 2, \ldots, 2^Q$.

Proof:

The code function, Φ_i, is just the limiting numerical value of the encoding of π_i followed by a number of zeros which tends toward infinity. The zeros do not change the fractional value of the input. Since $Z_i = 0$, the difference $\Phi_i - r_i$ will just be due to the value of the encoded zeros. But zeros, whether encoded or left as a remainder, have zero value.

Therefore,

$$\Phi_i = r_i \qquad \text{Q. E. D.}$$

Lemma 2:

If the remainder of π_i is all zeros, then $\Phi_i = r_i$, $i = 1, 2, \ldots, 2^Q$.

Proof:

The code function, Φ_i, will just be the limiting numerical value of the encoding of π_i followed by a number of zeros which tends toward infinity. As in Lemma 1, neither the value of q_i nor r_i changes as the number of zeros increases. Thus

$$\Phi_i = r_i \qquad \text{Q. E. D.}$$

Lemma 3:

If $Z_i = 0$ and $Z_{i+1} > 0$, then the remainder of π_{i+1} consists of only zeros, $i = 1, 2, \ldots, 2^Q - 1$.

Proof:

Assume the remainder of π_{i+1} has a non-zero digit. Then

$$r_i - \hat{r}_i = r_{i+1} - \hat{r}_{i+1} - 2^{-R_{i+1}}$$

and the prefixes are the same

$$\hat{\pi}_i = \hat{\pi}_{i+1}$$

Consequently, $Z_i > 0$ which is a contradiction. Q. E. D.

Lemma 4:

If $Z_i > 0$ and $Z_{i+1} > 0$ and if the remainder of π_i is not all ones then

$$\hat{\theta}_{i+1} = \hat{\theta}_i, \qquad i = 1, 2, \ldots, 2^Q-1$$

Proof:

$$q_{i+1} = q_i + 2^{-Q}$$

If the remainder of π_i is not all ones then the addition of 2^{-Q} to q_i will not carry into the prefix of q_i. Thus

$$\hat{\pi}_{i+1} = \hat{\pi}_i$$

and

$$\hat{\theta}_{i+1} = \hat{\theta}_i \qquad \text{Q. E. D.}$$

Lemma 5:

If $Z_i > 0$ and $Z_{i+1} = 0$ then the remainder of π_i must be all ones for $i = 1, 2, \ldots, 2^Q-1$.

Proof:

Assume the remainder of π_i is not all ones. Then adding 2^{-Q} to q_i would not carry a one into the prefix and

$$\hat{\pi}_{i+1} = \hat{\pi}_i$$

$$\hat{Q}_i = \hat{Q}_{i+1}$$

and

$$Z_{i+1} = Z_i > 0$$

This is a contradiction. Q. E. D.

Lemma 6:

If $Z_i > 0$, and the remainder of π_i is all ones, then

1) $\hat{r}_{i+1} = \hat{r}_i + 2^{-\hat{R}_i}$

2) either $Z_{i+1} = 0$ or the remainder of π_{i+1} is all zeros, and

3) $r_{i+1} = r_i + 2^{-R_i}$

for $i = 1, 2, \ldots, 2^Q - 1$.

Proof:

Obviously $\hat{R}_i > 0$, for otherwise π_i would be all ones and $i = 2^Q$ which is not an allowed index.

Assume the k^{th} word from the code input word set constitutes the input sequence prefix, $\hat{\pi}_i$. Adding 2^{-Q} to q_i carries a one into the $\hat{Q}^{th}$ place of q_i, i.e.,

$$q_{i+1} = \hat{q}_i + 2^{-\hat{Q}_i}$$

Now consider the condition of adjacency presented in the proof of Theorem 2.

$$\pi_i = \eta\ 0\ \zeta_k$$

and

$$\pi_{i+1} = \eta\ 1\ \mu_i$$

where

η = binary sequence of length m

ζ_k = sequence of k ones

μ_i = sequence of i zeros

Obviously, the index of the location of the right-most one in π_{i+1}, the $(m+1)^{st}$ location, is no larger than the length of π_i, m + 1 + k. Therefore, based on the results of Theorem 2, π_{i+1} consists of the $(k+1)^{st}$ word from the input word set possibly followed by all zero code words possibly followed by an all zero remainder. Therefore the output σ_{i+1} consists of the $(k+1)^{st}$ output word possibly followed by all zero code words possibly followed by an all zero remainder.

Therefore statements 1), 2), and 3) are true for the case where $\hat{\pi}_i$ consists of one code word.

If $\hat{\pi}_i$ consists of more than one word then the 2^{-Q} added to q_i changes the first word, going to the left from the remainder, which is not all ones and replaces it with the adjacent word in the input code word set. The symbols which follow are again all zero. Therefore statements 1), 2), and 3) again are true.

Q. E. D.

Lemma 7:

For a code of size N, the maximum and minimum values of R/Q are

$$\frac{1}{N-1} \leq R/Q \leq N-1$$

Proof:

For a code of size N, the maximum word length is N-1 and the minimum is 1. Therefore,

$$\frac{1}{N-1} \leq \frac{m_i}{\ell_i} \leq N-1, \qquad i = 1, 2, \ldots, N$$

and

$$\frac{1}{N-1} \leq R/Q \leq N-1 \qquad \text{Q. E. D.}$$

Proof of Theorem 7:

$$- \lim_{Q \to \infty} 1/Q \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (r_{i+1} - r_i)$$

$$= - \lim_{Q \to \infty} 1/Q \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 (\Phi(q_{i+1}) - \Phi(q_i))$$

$$- \lim_{Q \to \infty} T_Q$$

where

$$T_Q = 1/Q \sum_{i=1}^{2^Q} (F(q_{i+1}) - F(q_i)) \log_2 \frac{\Phi(q_{i+1}) - \Phi(q_i)}{r_{i+1} - r_i}$$

It will now be shown that

$$\lim_{Q \to \infty} T_Q = 0$$

The transition from π_i to π_{i+1} can be broken into four categories:

	π_i	π_{i+1}
1)	no remainder	no remainder
2)	no remainder	remainder
3)	remainder	remainder
4)	remainder	no remainder

For each category, bounds will be determined on E_i

$$E_i = \frac{\Phi_{i+1} - \Phi_i}{r_{i+1} - r_i}$$

$$i = 1, 2, \ldots, 2^Q - 1$$

where

$$\Phi_i = \Phi(q_i)$$

Category 1:

By Lemma 1

$$\Phi_i = r_i$$

$$\Phi_{i+1} = r_{i+1}$$

Thus

$$\Phi_{i+1} - \Phi_i = r_{i+1} - r_i$$

and

$$E_i = 1$$

Category 2:

By Lemma 3, the remainder of π_{i+1} is all zeros. By Lemma 2

$$\Phi_{i+1} = r_{i+1}$$

By Lemma 1

$$\Phi_i = r_i$$

$$\Phi_{i+1} - \Phi_i = r_{i+1} - r_i$$

and

$$E_i = 1$$

Category 3:

Assume the remainder of π_i is not all ones. Then by Lemma 4

$$\hat{\sigma}_{i+1} = \hat{\sigma}_i$$

$$\hat{r}_{i+1} = \hat{r}_i$$

and

$$r_{i+1} = r_i + 2^{-R_i}$$

By Theorem 3

$$1 \leq Z_i < \ell_{max} \leq N-1$$

so that

$$2^{-\hat{R}_i}\, 2^{-(N-1)} < r_{i+1} - r_i \leq 2^{-\hat{R}_i}\, 2^{-1} \qquad \text{(A-1)}$$

Define the first $\hat{R}_i$ digits in Φ_i as the prefix of Φ_i. It is now necessary to bound the maximum length of digits after the prefix of Φ_i which do not end in zero. This length will be denoted by V_i. To determine an upper bound on V_i, π_i will be suffixed by ℓ_{max} zeros. Then, by Lemma 7 and Theorem 3

$$V_i \leq (N-1)\,(Z_i + \ell_{max})$$

$$< (N-1)\ (2\ell_{max})$$

$$\leq 2(N-1)^2$$

$$2^{-\hat{R}_i}\ 2^{-2(N-1)^2} < 2^{-\hat{R}_i}\ 2^{-V_i} \qquad \text{(A-2)}$$

$$\leq \Phi_{i+1} - \Phi_i < 2^{-\hat{R}_i}$$

Combining Equations (A-1) and (A-2)

$$2^{-2(N-1)^2} + 1 < E_i < 2^{(N-1)}$$

Now assume the remainder of π_i is all ones. Then by Lemma 6

$$r_{i+1} - r_i = 2^{-R_i} = 2^{-\hat{R}_i}\ 2^{-Z_i}$$

and

$$2^{-\hat{R}_i}\ 2^{-(N-1)} < r_{i+1} - r_i \leq 2^{-\hat{R}_i}\ 2^{-1}$$

Bounds must now be determined for $\Phi_{i+1} - \Phi_i$. By Lemmas 6 and 2

$$\Phi_{i+1} = \hat{r}_{i+1} = \hat{r}_i + 2^{-\hat{R}_i}$$

$$< \Phi_i + 2^{-\hat{R}_i}$$

Also

$$\Phi_{i+1} \geq \Phi_i + 2^{-\hat{R}_i}\ 2^{-V_i}$$

$$2^{-\hat{R}_i}\, 2^{-2(N-1)^2} \leq \Phi_{i+1} - \Phi_i < 2^{-\hat{R}_i}$$

and

$$2^{-2(N-1)^2} + 1 \leq E_i < 2^{(N-1)}$$

Category 4:

By Lemma 5, all digits in the remainder of π_i must be ones. By Lemma 6

$$r_{i+1} = \Phi_{i+1} = r_i + 2^{-R_i}$$

$$= r_i + 2^{-\hat{R}_i}\, 2^{-Z_i}$$

so that

$$2^{-\hat{R}_i}\, 2^{-(N-1)} \leq r_{i+1} - r_i < 2^{-\hat{R}_i}\, 2^{-1}$$

From Equation (A-2)

$$2^{-\hat{R}_i}\, 2^{-2(N-1)^2} < \Phi_{i+1} - \Phi_i < 2^{-\hat{R}_i}$$

and

$$2^{-2(N-1)^2} + 1 < E_i < 2^{(N-1)}$$

Thus far, the transition from π_{2^Q} to $\pi_{2^{Q}+1}$ has not been treated. By definition

$$R_{2^{Q}+1} = \Phi_{2^{Q}+1} = 1.000$$

Obviously

$$r_{2^Q+1} - r_{2^Q} = 2^{-R_{2^Q}}$$

Therefore, if π_{2^Q} has no remainder, then the bounds of category 1) apply to E_{2^Q}. If π_{2^Q} has a remainder, the bounds of category 4) apply to E_{2^Q}. Therefore, for any category

$$2^{-2(N-1)^2+1} \leqq E_i < 2^{(N-1)}, \qquad i = 1, \ldots, 2^Q$$

Consequently

$$1/Q(-2(N-1)^2 + 1) \leqq T_Q < 1/Q\ (N-1)$$

and

$$\lim_{Q \to \infty} T_Q = 0$$

Q. E. D.

BIBLIOGRAPHY

1. Norman Abramson, Information Theory and Coding, New York: McGraw-Hill, 1963.

2. D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," Proceedings of the Institute of Radio Engineers, vol. 40, no. 10, pp. 1098-1101, September 1952.

3. E. N. Gilbert and E. F. Moore, "Variable Length Binary Encodings," Bell System Technical Journal, vol. 38, pp. 933-968, July 1959.

4. R. M. Karp, "Minimum-redundancy Coding for the Discrete Noiseless Channel," Institute of Radio Engineers Transactions on Information Theory, vol. IT-7, pp. 27-38, January 1961.

5. P. Billingsley, "On the Coding Theorem for the Noiseless Channel," Annals of Mathematical Statistics, vol. 32, no. 2, pp. 576-601, 1961.

6. R. S. Marcus, "Discrete Noiseless Coding," M.S. thesis in electrical engineering, Massachusetts Institute of Technology, Cambridge, Mass., 1957.

7. A. E. Laemmel and J. M. Brogan, "Coded Transmission of Information," Report R-325-53, PIB-261, Microwave Research Institute, Polytechnic Institute of Brooklyn, 1954.

8. N. M. Blachman, "Minimum-cost Encoding of Information," I. R. E. Trans. on Information Theory, vol. IT-3, pp. 139-149, March 1954.

9. Solomon W. Golomb, "Run-length Encodings," Institute of Electrical and Electronics Engineers Transactions on Information Theory, vol. IT-12, no. 3, pp. 399-401, July 1966.

10. Athanasios Papoulis, Probability, Random Variables, and Stochastic Processes, New York: McGraw-Hill, 1965.

11. A. E. Laemmel, "A General Class of Discrete Codes and Certain of Their Properties," Report R-459-55, PIB-389, Microwave Research Institute, Polytechnic Institute of Brooklyn, 1956.

12. J. G. Kemeny and J. L. Snell, Finite Markov Chains, Princeton, New Jersey: D. Van Nostrand, 1960.

VITA

Brian Parker Tunstall was born in Fort Bragg, North Carolina, on Jaunary 23, 1941. He is the son of Alfred M. and Virginia Parker Tunstall.

He attended public school in Tuscaloosa, Alabama, where he graduated from high school in 1958. In 1962 he received a B.S.E.E. from the Massachusetts Institute of Technology.

He worked for the Army Missile Command, Huntsville, Alabama, during the summers of 1961 and 1962. He held summer jobs at Bell Telephone Laboratories in Murray Hill, New Jersey, and at Lockheed-Georgia Company, Marietta, Georgia, in 1963 and 1964, respectively. He has taught introductory quantum mechanics and electronic circuits at Georgia Institute of Technology as a Graduate Teaching Assistant. He held a Schlumberger Fellowship from September, 1965, until June, 1966.